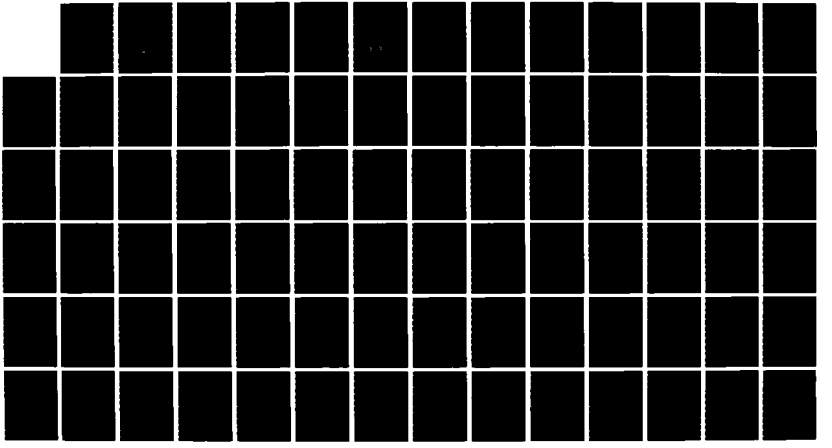
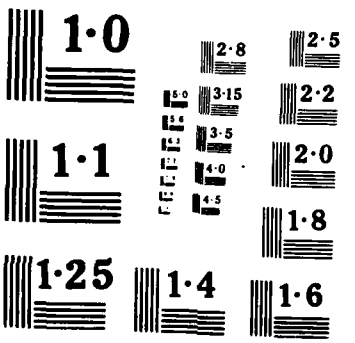
AD-A169 552 MMU (MANNED MANEUVERING UNIT) TASK SIMULATOR(U) AIR FORCE ACADEMY CO S ALFANO 15 JAN 86 USAFA-TR-86-2 1/1

UNCLASSIFIED F/G 22/1 NL




1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5

USAFA TR 86-2

(2)

MMU TASK SIMULATOR

SALVATORE ALFANO, MAJOR, USAF

JANUARY 1986

FINAL REPORT

DEAN OF THE FACULTY
UNITED STATES AIR FORCE ACADEMY
COLORADO SPRINGS, CO 80840

This research report is presented as a competent treatment of the subject, worthy of publication. The United States Air Force Academy vouches for the quality of the research, without necessarily endorsing the opinions and conclusions of the author.

This report has been cleared for open publication and/or public release by the appropriate Office of Information in accordance with AFM 190-1, AFR 12-30, and AFR 80-3. There is no objection to unlimited distribution of this report to the public at large, or by DTIC to the National Technical Information Service.

This research report has been reviewed and is approved for publication.

Thomas E. McCann

THOMAS E. McCANN, Lt Col, USAF
Director of Research and
Computer Based Education

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

AD-A169 552

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release. Unlimited distribution.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): USAFA TR 86-2	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Department of Astronautics; 6b. OFFICE SYMBOL (If applicable): DFAS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): U.S. Air Force Academy, Colorado Springs, CO 80840-5701	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification): MMU Task Simulator	

12. PERSONAL AUTHOR(S): SALVATORE ALFANO, Maj, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final report	FROM Jun 85 TO Jan 86	15 January 1986	78

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR	Simulation, MMU Manned Maneuvering Unit Task, Rendezvous, Proximity Ops
22	01		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper describes simplified mathematical models of the Manned Maneuvering Unit (MMU) used in the USAFA Proximity Operations Simulator for the VAX 11/780 and the Evans and Sutherland PS 300 computers. This simulator serves as a learning aid for cadets studying orbital dynamics and MMU mission planning and as a research platform for the Department of Astronautics.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
SALVATORE ALFANO, Maj, USAF	(303) 472-4110	DFAS

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE — UNCLASSIFIED

MMU Task Simulator

by

Major Salvatore Alfano

United States Air Force Academy
Department of Astronautics

January 15, 1986

ABSTRACT

This paper describes simplified mathematical models of the Manned Maneuvering Unit (MMU) used in the USAFA Proximity Operations Simulator for the VAX 11/780 and the Evans and Sutherland PS 300 computers. This simulator serves as a learning aid for cadets studying orbital dynamics and MMU mission planning and as a research platform for the Department of Astronautics.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

The Manned Maneuvering Unit (MMU) Proximity Operations Simulator is a nine degrees-of-freedom trajectory integrator (six degrees of freedom for the MMU and three degrees of freedom for the target) which generates digital and graphical data to describe relative motion of the MMU and a free-flying target. This motion is obtained by applying the Clohessy-Wiltshire equations for terminal rendezvous/docking with the earth modeled as a uniform sphere (~~Appendix~~ A) and aerodynamic forces ignored. MMU position relative to target is computed by a first-order Euler integrator which uses quaternions to define the rotational state (~~Appendix~~ B).

The target is modeled as a Space Transportation System (STS) Orbiter. The MMU is treated as a rigid body whose mass properties (gross weight, moments and products of inertia, and center of gravity location) are set within the program and remain constant for the entire simulation.

The initial state of the simulation is defined by the user. The program requires altitude and inclination of target orbit to determine proper viewing perspective and orbital dynamics. The user must also input MMU position relative to target. The program sets relative velocity and rotation rates to zero and defines initial MMU attitude such that the operator faces the target with his feet pointed towards the earth. The user also has the option of multiplying MMU responsiveness to facilitate proximity operations training.

After program initialization, user inputs are made through the hand controllers located in the MMU Mockup. The left controller is the Translational Hand Controller (THC) and is used for positioning. The right hand controller is the Rotational Hand Controller (RHC) and controls MMU attitude. Cross-coupling accelerations are not modeled in the simulation. The program reads inputs from the mockup and updates position and attitude every 40 milliseconds.

2. MMU ACCELERATION MODELING

Table 1 contains the steady-state MMU accelerations used in the simulation. Attitude deviations induced by cross coupling are not included. These accelerations are fixed within Subroutine THRUST and cannot be changed by the operator.

TABLE I EVA MASS PROPERTIES AND MMU PERFORMANCE

Weights

MMU with Full Propellant	148.4 kg	(327.1 lbs)
50th percentile male	77.1 kg	(170.0 lbs)
EMU - Medium	103.1 kg	(227.2 lbs)
EMU Helmet TV Camera (optional)	6.1 kg	(13.5 lbs)
EMU EVA Lights (optional)	2.9 kg	(6.5 lbs)
EVA Mission	337.6 kg	(744.3 lbs)

Moments of Inertia

I_x = 56.4 kg-m^2 (41.6 slug-ft^2)

I_y = 57.9 kg-m^2 (42.7 slug-ft^2)

I_z = 32.7 kg-m^2 (24.1 slug-ft^2)

Products of Inertia

P_{xy} = 0.07 kg-m^2 (0.05 slug-ft^2)

P_{xz} = 2.28 kg-m^2 (1.68 slug-ft^2)

P_{yz} = -0.04 kg-m^2 (-0.03 slug-ft^2)

Acceleration Rates

Linear: a = 0.09 m/sec^2 (0.294 ft/sec^2)

Rotational: Pitch = 7.98 deg/sec^2

Yaw = 9.06 deg/sec^2

Roll = 8.19 deg/sec^2

Available Propulsive Forces and Torques

Linear: F = 30.25 Newtons (6.80 lb)

Rotational: Pitch = 0.67 N-m (5.95 ft-lb)

Yaw = 0.21 N-m (1.91 ft-lb)

Roll = 0.67 N-m (5.95 ft-lb)

(Reproduced from Reference 1)

3. DETERMINING POSITION AND ATTITUDE

The MMU translational accelerations are added to the orbital drift accelerations. These accelerations are then integrated twice to yield velocity and position. Attitude is determined by applying rotational accelerations to the present quaternions.

The Clohessy-Wiltshire equations for terminal rendezvous/docking are used to model orbital drift. These are linearized equations of motion for an interceptor vehicle relative to a target vehicle in a circular orbit with Keplerian motion.

$$\ddot{x} = f_x' - 2\omega\dot{y} \tag{1}$$

$$\ddot{y} = f_y' + 3\omega^2 y + 2\omega\dot{x} \tag{2}$$

$$\ddot{z} = f_z' - \omega^2 z \tag{3}$$

where f_x', f_y', and f_z' are the MMU translational acceleration components (due to thrust), and ω is the rotation rate of the target about the planet.

The target frame is a right handed orthogonal system where x is the direction of target velocity, y is the zenith direction (along target radius vector), and z is out of orbital plane (opposite the angular momentum vector). For further explanation and derivation of Equations 9-11 refer to Appendix A.

The translational accelerations due to drift and thrust are summed in Subroutine THRUST and then integrated in Subroutine LINTEG. LINTEG is a first order Euler integrator which was chosen for fast computational speed (needed in real time simulation) in light of the fact that accelerations, and hence error, will be small.

The rotational accelerations are transformed from the body frame to the reference (target) frame by Subroutine BTOR. These accelerations are then used by Subroutine ROTATE to determine a new attitude quaternion and transformation matrix. The equations used in Subroutine ROTATE are included in Appendix B.

4. GENERATION OF VISUAL DISPLAY

The simulator uses an Evans and Sutherland PS300 computer for visual display. Three dimensional object data are loaded and stored in the mass memory of the PS300 at the beginning of the program. These data are then rotated, translated, and displayed repeatedly as commanded by the main program from the VAX 11/780.

Subroutine PS300 is responsible for loading the object data, known as vector lists, and providing a hierarchy of rotation, translation and viewing commands for later input by Subroutine LOOK. The vector lists consist of a spherical outline of the earth's continents, a star sphere, a circular horizon, and STS Orbiter. The reference coordinate system is the Clohessy-Wiltshire system, centered at the target and described in detail in Appendix A.

The earth's vector list is scaled at one distance unit (DU) and rotated about its axis at the rate of 15 degrees per hour. The earth is then inclined and counter-rotated a rate equivalent to the angular rate of the target orbit. Finally, it is translated in the -Y direction an amount equal to its radius plus target altitude. To complete the earth picture a horizon circle is added and the earth vector list is clipped to prevent viewing beyond the horizon.

The star vector list is triplicated and rotated 90 degrees about each axis to create a unit sphere of stars. This representation does not reflect true star positions. The sphere is scaled up by a factor of one earth radius plus twice target altitude and set counter-rotating (as was the earth). The star sphere is then translated in the -Y direction an amount equal to one DU plus target altitude and clipped so no stars appear below the horizon.

The Orbiter vector list is scaled and placed at the center of the reference coordinate system. It is not clipped and remains at the origin throughout the simulation. Run time is displayed in the upper left hand corner.

Subroutine LOOK uses MMU position and attitude data relative to the target to continuously update rotation, translation, and viewing of all the predefined vector lists. Current position and attitude are used to generate three viewing vectors in the reference (target) frame: AT, FROM, and UP. AT is the line of sight vector, FROM is the position vector, and UP is the overhead vector (perpendicular to

AT). The Evans and Sutherland PS300 uses these viewing vectors to scale, translate, and re-orient the stored images for perspective viewing. Figure 1 shows the relationship of the stored images to each other and how they are viewed given AT, FROM and UP vectors.

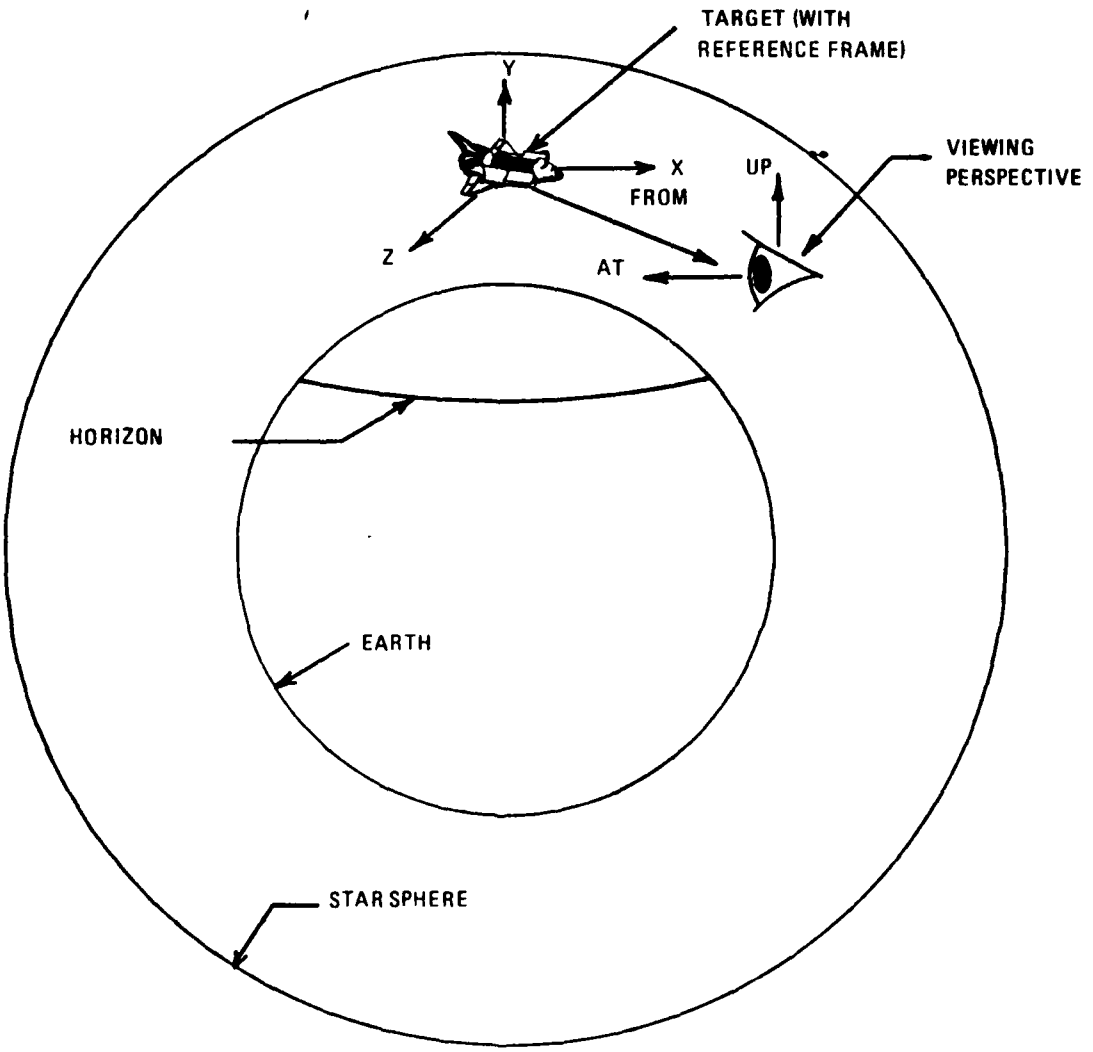


Figure 1. Visual Display Layout

REFERENCES

1. MMU-SE-17-46, "Manned Maneuvering Unit User's Guide", Martin Marietta, April 1982

2. Astro 451 Course Handout, Astro 451, Course Readings #2, Fall 1984, Chapter F, USAF Academy, Colorado, 1984

3. Hamilton Standard Paper, Strapdown Attitude Determination Using Quaternions, Hamilton Standard Division of United Technologies, Undated

APPENDIX A

(This appendix was taken wholly from Reference 2)

TERMINAL RENDEZVOUS/DOCKING

This appendix develops the equations of motion for an interceptor vehicle relative to a target vehicle in orbit about a central body when the range between vehicles is less than 8 km (5 miles).

The following approach will be taken to solve this engineering problem.

1. Establish a coordinate system:

   a. Origin at target vehicle

   b. Orthogonal right-handed system

   c. x -- In the local horizontal, in direction of target vehicle velocity vector

   d. y -- In zenith direction (along target vehicle position vector R)

   e. $\bar{R}$ -- Vector to target in fixed frame

   f. $\bar{r}$ -- Vector to interceptor in fixed frame

   g. $\bar{p} = \bar{r} - \bar{R}$ -- Position of the interceptor relative to the target.

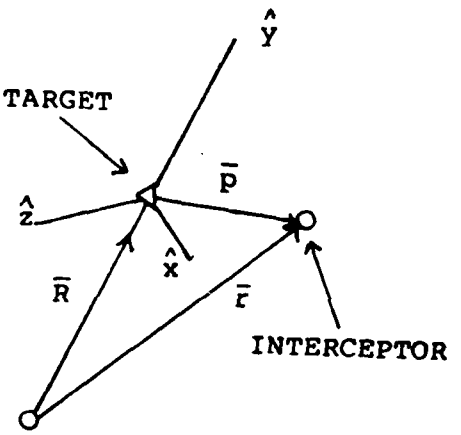


Figure 2

2. Apply $\Sigma \bar{F}_{ext} = \frac{d}{dt}(\bar{m}v)$ in the rotating coordinate system:

a. This is a double application of the law of coriolis for derivatives in a rotating frame. (If you can't derive this, see BMW pp. 92-93.)

$$\frac{d^2(\ )}{dt^2}_F = \frac{d^2(\ )}{dt^2}_R + \dot{\bar{\omega}} \times (\ ) + \bar{\omega} \times \bar{\omega} \times (\ ) + 2\,\bar{\omega} \times \frac{d(\ )}{dt}_R$$

b. When ( ) is the position of the interceptor vehicle, $(\bar{r})$, then

$$\left(\frac{d^2\bar{r}}{dt^2}\right)_F = \frac{\Sigma \bar{F}_{ext}}{m} = \bar{f} = \ddot{\bar{r}}_R + \dot{\bar{\omega}} \times \bar{r} + \bar{\omega} \times \bar{\omega} \times \bar{r} + 2\,\bar{\omega} \times \dot{\bar{r}}_R$$

c. Noting the following relationships

(1) $\bar{\omega} = -\omega\,\hat{z}$

(2) $\dot{\bar{\omega}} = -\dot{\omega}\,\hat{z}$

(3) $\bar{p} = x\,\hat{x} + y\,\hat{y} + z\,\hat{z}$

(4) $\dot{\bar{p}}_R = \dot{x}\,\hat{x} + \dot{y}\,\hat{y} + \dot{z}\,\hat{z}$

(5) $\ddot{\bar{p}}_R = \ddot{x}\,\hat{x} + \ddot{y}\,\hat{y} + \ddot{z}\,\hat{z}$

(6) $\bar{R} = R\,\hat{y}$

(7) $\dot{\bar{R}}_R = \dot{R}\,\hat{y}$

(8) $\ddot{\bar{R}}_R = \ddot{R}\,\hat{y}$

(9) $\bar{r} = x\,\hat{x} + (R + y)\,\hat{y} + z\,\hat{z}$

(10) $\dot{\bar{r}}_R = \dot{x}\,\hat{x} + (\dot{R} + \dot{y})\,\hat{y} + \dot{z}\,\hat{z}$

(11) $\ddot{\bar{r}}_R = \ddot{x}\,\hat{x} + (\ddot{R} + \ddot{y})\,\hat{y} + \ddot{z}\,\hat{z}$

d. Developing the required cross products

$$\dot{\bar{\omega}} \times \bar{r} = \dot{\omega}(R + y)\,\hat{x} - \dot{\omega}x\,\hat{y}$$

$$\bar{\omega} \times \dot{\bar{r}} = \omega(\dot{R} + \dot{y})\,\hat{x} - \omega\dot{x}\,\hat{y}$$

$$\bar{\omega} \times \bar{\omega} \times \bar{r} = -\omega^2 x\,\hat{x} - \omega^2(R + y)\,\hat{y}$$

e. Now substituting from c and d into the component form of the vector equation developed in 2.b, we obtain

$$f_x = \ddot{x} + \dot{\omega}(R + y) - \omega^2 x + 2\omega(\dot{R} + \dot{y})$$

$$f_y = \ddot{R} + \ddot{y} - \dot{\omega}x - \omega^2(R + y) - 2\omega\dot{x}$$

$$f_z = \ddot{z}$$

f. Now, let $\bar{f}'$ be the specific force other than the gravitational attraction of the central body (i.e., drag, thrust, solar pressure, etc.)

$$\bar{f} = \bar{f}' + \bar{f}_g = \bar{f}' - \frac{\mu r}{r^3}\hat{r}$$

OR

$$f_x = f_x' - \frac{\mu x}{r^3}$$

$$f_y = f_y' - \frac{\mu(R + y)}{r^3}$$

$$f_z = f_z' - \frac{\mu z}{r^3}$$

g. Then the relative motion is described by

$$\ddot{x} = f_x' - \dot{\omega}(R + y) + \omega^2 x - 2\omega(\dot{R} + \dot{y}) - \frac{\mu x}{r^3}$$

$$\ddot{y} = f_y' - \ddot{R} + \dot{\omega}x + \omega^2(R + y) + 2\omega\dot{x} - \frac{\mu(R + y)}{r^3}$$

$$\ddot{z} = f_z' - \frac{\mu z}{r^3} \quad \text{where} \quad r^3 = [x^2 + (R + y)^2 + z^2]^{3/2}$$

THESE ARE VERY NON-LINEAR, BUT EXACT EQUATIONS OF MOTION.

3. Linearize the equations to find an analytic solution:

$$r^3 = R^3\left[\left(\frac{x}{R}\right)^2 + \left(1 + \frac{y}{R}\right)^2 + \left(\frac{z}{R}\right)^2\right]^{3/2}$$

$$= R^3\left[1 + \frac{2y}{R} + \left(\frac{x}{R}\right)^2 + \left(\frac{y}{R}\right)^2 + \left(\frac{z}{R}\right)^2\right]^{3/2}$$

a. Since x, y and z are small compared with R,

$$r^3 \simeq R^3\left(1 + \frac{2y}{R}\right)^{3/2}$$

b. Then the last terms in the equations of 2.g. take on the form

$$-\frac{\mu}{r^3} \approx -\frac{\mu}{R^3\left(1+\frac{2y}{R}\right)^{3/2}}$$

c. Since the term $(\frac{y}{R})$ is small (less than 0.00125), use the binominal expansion to begin to simplify these equations

$$(1+\epsilon)^{-3/2} = 1 - \frac{3}{2}\epsilon + \frac{15}{8}\epsilon^2 - \ldots$$

d. Then, substituting into 2g and neglecting the small high order terms,

$$-\frac{\mu x}{R^3\left(1+\frac{2y}{R}\right)^{-3/2}} \approx -\frac{\mu x}{R^3} + \frac{3\mu xy}{R^4} \quad \text{, etc.}$$

So that

$$\ddot{x} = f_x' - \dot{\omega}(R+y) + \omega^2 x - 2\omega(\dot{R}+\dot{y}) - \frac{\mu x}{R^3} + \frac{3\mu xy}{R^4}$$

$$\ddot{y} = f_y' - \ddot{R} + \dot{\omega}x + \omega^2(R+y) + 2\omega\dot{x} - \frac{\mu}{R^2} + \frac{2\mu y}{R^3} + \frac{3\mu y^2}{R^4}$$

$$\ddot{z} = f_z' - \frac{\mu z}{R^3} + \frac{3\mu zy}{R^4}$$

e. The only nonlinear terms are the last ones in the equations. These terms are smaller than the proceeding terms by $(\frac{y}{R})$. Neglecting these terms results in

$$\ddot{x} = f_x' - \dot{\omega}(R+y) + \left(\omega^2 - \frac{\mu}{R^3}\right)x - 2\omega(\dot{R}+\dot{y})$$

$$\ddot{y} = f_y' - \ddot{R} + \dot{\omega}x + \left(\omega^2 - \frac{\mu}{R^3}\right)R + \left(\omega^2 + \frac{2\mu}{R^3}\right)y + 2\omega\dot{x}$$

$$\ddot{z} = f_z' - \frac{\mu}{R^3} z$$

4. Now ASSUME the target vehicle is in a CIRCULAR orbit.

   a. Then

$$\omega = \frac{V_{CS}}{R} = \sqrt{\frac{\mu}{R^3}} \quad , \quad \omega^2 = \frac{\mu}{R^3} \quad , \quad \dot{\omega} = 0 \quad , \quad \dot{R} \quad \text{and} \quad \ddot{R} = 0$$

   b. The equations of motion become

$$\ddot{x} = f_x' - 2\omega\dot{y}$$

$$\ddot{y} = f_y' + 3\omega^2 y + 2\omega\dot{x}$$

$$\ddot{z} = f_z' - \omega^2 z$$

THESE ARE THE LINEARIZED EQUATIONS OF MOTION FOR AN INTERCEPTOR VEHICLE RELATIVE TO A TARGET VEHICLE IN A CIRCULAR ORBIT WITH KEPLARIAN MOTION.

APPENDIX B

(This appendix was taken wholly from Reference 3)

ATTITUDE DETERMINATION USING QUATERNIONS

The quaternion q describing the orientation of a body with respect to a reference coordinate frame may be found by integrating the quaternion differential equation:

$$\dot{q} = q \frac{\omega}{2} \tag{1}$$

where

$$q = \hat{i}q_1 + \hat{j}q_2 + \hat{k}q_3 + q_4$$

$$\omega = \hat{i}\omega_X + \hat{j}\omega_Y + \hat{k}\omega_Z$$

= rate of rotation of body with respect to the reference frame (in body coordinate frame).

Expansion of (1) yields the following form:

$$\frac{d}{dt}\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & \omega_Z & -\omega_Y & \omega_X \\ -\omega_Z & 0 & \omega_X & \omega_Y \\ \omega_Y & -\omega_X & 0 & \omega_Z \\ -\omega_X & -\omega_Y & -\omega_Z & 0 \end{bmatrix} \cdot \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{2}$$

The quaternion components in the body (X, Y, Z) coordinate frame are

$$q_1 = \frac{u_X}{u} \sin \frac{ut}{2}$$

$$q_2 = \frac{u_Y}{u} \sin \frac{ut}{2} \qquad (3)$$

$$q_3 = \frac{u_Z}{u} \sin \frac{ut}{2}$$

$$q_4 = \cos \frac{ut}{2}$$

where

$$u = \sqrt{u_X^2 + u_Y^2 + u_Z^2}$$

A vector is transformed from body to reference coordinates by

$$\bar{X}^R = q\bar{X}^B q^* \qquad (4)$$

where

$$q^* = -\hat{i}q_1 - \hat{j}q_2 - \hat{k}q_3 + q_4$$

$$= \text{conjugate of } q$$

$$qq^* = q^* q = 1$$

The equivalent equation in vector-matrix notation is

$$\bar{X}^R = \left[T_B^R\right] \bar{X}^B \qquad (5)$$

If (4) and (5) are expanded and compared, it is seen that

$$\left[T_B^R\right] = \begin{bmatrix} q_1^2-q_2^2-q_3^2+q_4^2 & 2(q_1q_2-q_3q_4) & 2(q_1q_3+q_2q_4) \\ 2(q_1q_2+q_3q_4) & -q_1^2+q_2^2-q_3^2+q_4^2 & 2(q_2q_3-q_1q_4) \\ 2(q_1q_3-q_2q_4) & 2(q_2q_3+q_1q_4) & -q_1^2-q_2^2+q_3^2+q_4^2 \end{bmatrix} \quad (6)$$

From (6), the quaternion components, expressed as functions of the matrix elements, are

$$\begin{aligned} q_1 &= \frac{1}{4q_4}\,(T_{32} - T_{23}) \\ q_2 &= \frac{1}{4q_4}\,(T_{13} - T_{31}) \\ q_3 &= \frac{1}{4q_4}\,(T_{21} - T_{12}) \\ q_4 &= \frac{1}{2}\sqrt{1 + T_{11} + T_{22} + T_{33}} \end{aligned} \quad (7)$$

APPENDIX C

PROGRAM LISTING

```
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C    PRODUCED BY THE UNITED STATES AIR FORCE ACADEMY
C    DEPARTMENT OF ASTRONAUTICS AND IS NOT PROTECTED
C    BY COPYRIGHT.              (17 U.S.C. SECTION 105)
C        PROGRAM IS BASED ON WORK BY USAFA/DFAS.
C    AUTOVON 259-4110           COMMERCIAL 303-472-4110
C^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
C      EXECUTES THRUST COMMANDS FOR PROXIMITY OPERATIONS
C      OF MANNED MANEUVERING UNIT.
C      QUESTIONS ?????  CONTACT CAPT ALFANO, DFAS
C
C      LINK AS FOLLOWS
C       LINK MMUSIM,[ALFANO]ERROR,LPAIO,GSRF/LIB
C
       INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'
C
       REAL X(6),T(3,3),Q(4)
       REAL WB(3),ALT,AB(3),AR(3),ANG(3)
       REAL DELTAT,UP(3),FUDGE,STOPIT
       REAL DELT1,DELT2,INCLIN,TIME
       REAL RANGE,OMEGA,OLDX(3)
C
       INTEGER*4 SYS$SETEF,SYS$WAITFR,CHAN
       INTEGER*2 IBUF(193),IOSB(4)
       INTEGER C(12),I,J,DC(12),RESTOP,OVHD(12)
       INTEGER NLOOPS,RESET,FLAG
       CHARACTER*16 TIMBUF
       INTEGER*4 TIMADR
C
C         AB(3) - ACCELERATION (BODY FRAME)
C         ALT - ALTITUDE OF TARGET (KM)
C         ANG(3) - ANGULAR ACCELERATION (BODY FRAME)
C         AR(3) - ACCELERATION (REF FRAME)
C         CHAN - CHANNEL FOR PS300 PHYSICAL I/O
C         DELT1,2 - TIME COUNTERS TO SEE IF LOOP EXCEEDS .04 SEC
C         DELTAT - TIME STEP
C         FLAG - FLAG TO READ SHUTTLE MOCKUP
C         FUDGE - FUDGE FACTOR TO CHANGE MMU RESPONSIVENESS
C         IBUF - BUFFER FOR PS300 PHYSICAL I/O
C         INCLIN - INCLINATION OF TARGET ORBIT (DEG)
C         IOSB - ARRAY FOR PS300 PHYSICAL I/O
C         NLOOPS - LOOP COUNTER
C         OLDX(3) - ORIGINAL MMU POSITION FROM TARGET IN C-W FRAME
C         OMEGA - ANGULAR RATE OF TARGET ABOUT EARTH (RAD/SEC)
C         OVHD(12) - ARRAY FOR COMMANDS FROM SHUTTLE MOCKUP
C         Q(4) - QUATERNION
C         RANGE - RANGE FROM TARGET
C         RESET - COMMAND FLAG TO START OVER FROM ORIGINAL POSITION
C         RESTOP - COUNTER TO STOP PROGRAM IF RESET HELD FOR 1 SEC
C         STOPIT - MAX RUN TIME (MINUTES)
C         T(3,3) - TRANSITION MATRIX (BODY TO REFERENCE FRAME)
C         TIME - TIME (SEC)
C         TIMBUF,TIMADR - USED TO SET ITERATIVE LOOP TO DELTAT
C         UP(3) - UP VECTOR FOR PS300 AND ALIGNMENT SUBROUTINE
C         WB(3) - ROTATION RATE (BODY FRAME)
C         X(6) - MMU POSITION/VELOCITY FROM TARGET IN C-W FRAME
C
C         ASSIGNMENT OF THRUST COMMANDS
C
```

```
C          C( ) - COMMAND ARRAY (READ FROM THC/RHC)
C
C          C(1) - +X
C          C(2) - -X
C          C(3) - +Y
C          C(4) - -Y
C          C(5) - +Z
C          C(6) - -Z
C          C(7) - +ROL
C          C(8) - -ROL
C          C(9) - +PCH
C          C(10) - -PCH
C          C(11) - +YAW
C          C(12) - -YAW
C
C          1=ON  0=OFF
C
C          INITIALIZATIONS
C
      DATA TIMBUF/'0000 00:00:00.12'/
      DATA (C(I),I=1,12)/12*0/
      DATA (X(I),I=1,6)/6*0/
C
C        SET TIMER
C
      CALL SYS$BINTIM(TIMBUF,TIMADR)
C
C          READ IN MAX RUN TIME
C
      PRINT *,' HOW MANY MINUTES DO YOU WISH TO RUN ?'
      PRINT *,' (IF OVER 10, PLEASE COORDINATE WITH SEILER)'
      READ *,STOPIT
C
C          READ IN TARGET ALTITUDE
C
      PRINT *,' ENTER ALTITUDE OF TARGET (KM)'
      READ *,ALT
C
C          COMPUTE ROTATION RATE OF TARGET ABOUT EARTH
C
      OMEGA=SQRT(398600.8/(ALT+6378.135)**3)
C
C          READ IN INCLINATION OF TARGET ORBIT
C
      PRINT *,' ENTER INCLINATION OF TARGET ORBIT (DEG)'
      READ *,INCLIN
C
C          READ IN FUDGE FACTOR
C
      PRINT *,'BY WHAT FACTOR WOULD YOU LIKE TO INCREASE'
      PRINT *,'MMU RESPONSIVENESS ?'
      READ *,FUDGE
C
C          READ IN INITIAL MMU POSITION
C
      PRINT *,' ENTER COORDINATES OF MMU FROM TARGET'
      PRINT *,' X,Y,Z CLOHESSY-WILTSHIRE FRAME (FEET)'
      READ *,OLDX(1),OLDX(2),OLDX(3)
C
C        ZERO VELOCITY AND ANGULAR MOTION ARE ASSUMED FOR MMU.
```

```
C         MMU ALIGNMENT IS INITIALIZED SUCH THAT TARGET
C         STARTS IN CENTER OF SCREEN (X IN BODY FRAME)
C
C         INITIALIZE PS300 DISPLAY
C         INITIALIZE DR-11K HAND CONTROLLER BOARD
C         READ MEMORY ADDRESS LOCATIONS IN PS300
C
        CALL PS300(ALT,INCLIN)
        CALL IOBUFF(1,DC,RESET)
        CALL INITBUF(IBUF,CHAN,IOSB)
C
C         SET EVENT FLAG FOR WAIT COMMAND
C
        CALL SYS$SETEF(7)
C
C         RESET BEGINS HERE
C
135     CONTINUE
C
        WB(1)=0.0
        WB(2)=0.0
        WB(3)=0.0
        DELTAT=.04
C
        UP(1)=0.0001
        UP(2)=1.0
        UP(3)=0.0001
C
        X(1)=OLDX(1)
        X(2)=OLDX(2)
        X(3)=OLDX(3)
        X(4)=0.0
        X(5)=0.0
        X(6)=0.0
C
        DO 175 I=1,12
          C(I)=0
          DC(I)=0
175     CONTINUE
C
C           ALIGN MMU SO TARGET IS CENTERED IN WINDOW
C
        CALL ALIGN(X(1),X(2),X(3),T,Q,UP)
C
C           INITIALIZE COUNTERS
C
        NLOOPS=-1
        DELT1=SECNDS(0.)
        RESET=0
        FLAG=0
C
C           ITERATIVE LOOP BEGINS HERE
C
450     CONTINUE
C
C           UPDATE COUNTERS
C
        NLOOPS=NLOOPS+1
        TIME=NLOOPS*DELTAT
        FLAG=FLAG+1
```

```
C
C          APPLY COMMANDS (C)
C
      CALL THRUST(AB,ANG,C)
C
C          APPLY FUDGE FACTOR
C
      AB(1)=FUDGE*AB(1)
      AB(2)=FUDGE*AB(2)
      AB(3)=FUDGE*AB(3)
C
C          TRANSLATE BODY ACCELERATIONS (AB) TO REF FRAME (AR)
C
      CALL BTOR(T,AB,AR)
C
C          APPLY ACCELERATIONS TO DETERMINE POSITION (X(1-3))
C          AND VELOCITY (X(4-6)) OF MMU RELATIVE TO TARGET
C          USING CLOHESSY-WILTSHIRE EQUATIONS.
C
      CALL LINTEG(X,OMEGA,AR,DELTAT)
C
C          APPLY ANGULAR ACCELERATIONS TO FIGURE QUATERNIONS AND NEW T MATRIX
C
      CALL ROTATE(ANG,DELTAT,WB,Q,T)
C
C          DETERMINE TARGET POSITION AND ATTITUDE
C          AS SEEN DIRECTLY AHEAD.
C          DISPLAY ON THE PS300
C
      CALL LOOK(T,X(1),X(2),X(3),OMEGA,TIME,IBUF,CHAN,IOSB)
C
C          READ THRUST COMMANDS
C
      IF (FLAG .GE. 3) THEN
        CALL IOBUFF(2,OVHD,RESET)
        CALL SYS$WAITFR(7)
        CALL SYS$SETIMR(7,TIMADR,,)
C
C          IF RESET HELD FOR 25 ITERATIONS, STOP PROGRAM
C          IF RESET COMMANDED GO TO 135
C
        IF (RESET .EQ. 1) THEN
          RESTOP=RESTOP+1
          IF (RESTOP .GE. 15) GOTO 999
          GOTO 135
         ELSE
          RESTOP=0
        ENDIF
        FLAG=0
C
C          CONVERT FROM SHUTTLE CONTROLLERS TO MMU
C
        DC(1)=OVHD(2)
        DC(2)=OVHD(1)
        DC(3)=OVHD(4)
        DC(4)=OVHD(3)
        DC(5)=OVHD(5)
        DC(6)=OVHD(6)
        DC(7)=OVHD(8)
        DC(8)=OVHD(7)
```

```
          DC(9)=OVHD(10)
          DC(10)=OVHD(9)
          DC(11)=OVHD(11)
          DC(12)=OVHD(12)
C
        ENDIF
C
C          RESTORE THRUST COMMANDS
C
        DO 645 I=1,12
          C(I)=DC(I)
645     CONTINUE
C
C          RETURN TO 450 FOR NEXT ITERATION
C
        IF (NLOOPS .LT. (1500*STOPIT)) GOTO 450
C
C          COMPUTE AVERAGE LOOP TIME
C
        DELT2=SECNDS(DELT1)
        PRINT *,' AVERAGE LOOP TIME = ',DELT2/NLOOPS,' SECONDS'
C
C
999     CONTINUE
C
        END
C
C
C
C
        SUBROUTINE ALIGN(X,Y,Z,T,Q,UP)
C
C       COMPUTES BODY AXIS COMPONENTS IN REFERENCE FRAME.
C       -XBOD IS THE POSITION VECTOR.  ZBOD AND YBOD
C       COMPLETE THE RIGHT HANDED SYSTEM.
C       DETERMINES QUATERNION (Q) AND TRANSFORMATION
C       MATRIX (T).
C
        REAL X,Y,Z,T(3,3),Q(4),UP(3),CONST
        REAL XBOD(3),YBOD(3),ZBOD(3)
C
        XBOD(1)=-X
        XBOD(2)=-Y
        XBOD(3)=-Z
C
C        DO NOT ALLOW XBOD TO LIE DIRECTLY ON AN AXIS.
C        THIS IS DONE TO PREVENT QUATERNION AMBIGUITIES.
C
        IF (ABS(XBOD(1)) .LT. .001) XBOD(1)=.001
        IF (ABS(XBOD(2)) .LT. .001) XBOD(2)=.001
        IF (ABS(XBOD(3)) .LT. .001) XBOD(3)=.001
C
C        COMPUTE TRANSFORMATION MATRIX
C
        CALL UNITIZE(XBOD)
        CALL CROSS(YBOD,XBOD,UP)
        CALL UNITIZE(YBOD)
        CALL CROSS(ZBOD,XBOD,YBOD)
        CALL UNITIZE(ZBOD)
C
```

```
      T(1,1)=XBOD(1)
      T(1,2)=YBOD(1)
      T(1,3)=ZBOD(1)
      T(2,1)=XBOD(2)
      T(2,2)=YBOD(2)
      T(2,3)=ZBOD(2)
      T(3,1)=XBOD(3)
      T(3,2)=YBOD(3)
      T(3,3)=ZBOD(3)
C
C       COMPUTE QUATERNIONS
C
      Q(4)=1.0+T(1,1)+T(2,2)+T(3,3)
      IF (Q(4) .LT. .1E-30) Q(4)=.1E-30
      Q(4)=SQRT(Q(4))/2.0
      Q(1)=(T(3,2)-T(2,3))/(4.0*Q(4))
      Q(2)=(T(1,3)-T(3,1))/(4.0*Q(4))
      Q(3)=(T(2,1)-T(1,2))/(4.0*Q(4))
C
C       NORMALIZE QUATERNIONS
C
      CONST=SQRT(Q(1)*Q(1)+Q(2)*Q(2)+Q(3)*Q(3)+Q(4)*Q(4))
      Q(1)=Q(1)/CONST
      Q(2)=Q(2)/CONST
      Q(3)=Q(3)/CONST
      Q(4)=Q(4)/CONST
C
      END
C
C
C
C
      SUBROUTINE THRUST(A,ANG,C)
C
C     A - ACCEL X,Y,Z (FPS2,BODY FRAME)
C     ANG - ANGULAR ACCEL X,Y,Z (RAD/S2,BODY FRAME)
C
      REAL A(3),ANG(3)
      INTEGER I,J,C(12)
C
C
C      INTIALIZE A AND ANG
C
      DO 20 I=1,3
        ANG(I)=0.0
        A(I)=0.0
20    CONTINUE
C
C      DETERMINE ACCELERATIONS DUE TO COMMANDED THRUST
C
      IF (C(1) .EQ. 1) A(1)=.323
      IF (C(2) .EQ. 1) A(1)=-.323
      IF (C(3) .EQ. 1) A(2)=.323
      IF (C(4) .EQ. 1) A(2)=-.323
      IF (C(5) .EQ. 1) A(3)=.323
      IF (C(6) .EQ. 1) A(3)=-.323
      IF (C(7) .EQ. 1) ANG(1)=.142942
      IF (C(8) .EQ. 1) ANG(1)=-.142942
      IF (C(9) .EQ. 1) ANG(2)=.13927
      IF (C(10) .EQ. 1) ANG(2)=-.13927
```

```
      IF (C(11) .EQ. 1) ANG(3)=.158127
      IF (C(12) .EQ. 1) ANG(3)=-.158127
C
      END
C
C
C
C
      SUBROUTINE LINTEG(X,OMEGA,A,DELTAT)
C
C     THIS IS A FIRST ORDER INTEGRATION SCHEME FOR TRANSLATIONAL
C     ACCELERATIONS IN THE REF (C-W) FRAME USING C-W EQUATIONS
C
      REAL X(6),XDOT(6),A(3),OMEGA,DELTAT
      INTEGER J
C
C       A(1-3) - X,Y,Z ACCELERATION
C       X(1-3) - X,Y,Z POSITION
C       X(4-6) - X,Y,Z VELOCITY
C       DELTAT - TIME STEP
C       OMEGA - ANGULAR RATE OF TARGET ABOUT EARTH
C
      XDOT(1)=X(4)
      XDOT(2)=X(5)
      XDOT(3)=X(6)
C
C       BELOW ARE THE LINEARIZED EQUATIONS OF MOTION FOR AN INTERCEPT
C       VEHICLE RELATIVE TO A TARGET VEHICLE IN A CIRCULAR ORBIT
C       WITH KEPLARIAN MOTION
C
      XDOT(4)=A(1)-2.0*OMEGA*X(5)
      XDOT(5)=A(2)+3.0*OMEGA*OMEGA*X(2)+2.0*OMEGA*X(4)
      XDOT(6)=A(3)-X(3)*OMEGA*OMEGA
C
      DO 100 J=1,6
        X(J)=X(J)+DELTAT*XDOT(J)
100   CONTINUE
C
        END
C
C
C
      SUBROUTINE BTOR(T,BOD,REF)
C
C     TRANSFORMS FROM BOD TO REF FRAME GIVEN TRANSFORMATION MATRIX T
C
      REAL T(3,3),BOD(3),REF(3)
C
      REF(1)=BOD(1)*T(1,1)+BOD(2)*T(1,2)+BOD(3)*T(1,3)
      REF(2)=BOD(1)*T(2,1)+BOD(2)*T(2,2)+BOD(3)*T(2,3)
      REF(3)=BOD(1)*T(3,1)+BOD(2)*T(3,2)+BOD(3)*T(3,3)
C
      END
C
C
C
C
      SUBROUTINE RTOB(T,REF,BOD)
C
```

```
C       TRANSFORMS FROM REF TO BOD FRAME GIVEN TRANSFORMATION MATRIX T
C
        REAL T(3,3),REF(3),BOD(3)
C
        BOD(1)=REF(1)*T(1,1)+REF(2)*T(2,1)+REF(3)*T(3,1)
        BOD(2)=REF(1)*T(1,2)+REF(2)*T(2,2)+REF(3)*T(3,2)
        BOD(3)=REF(1)*T(1,3)+REF(2)*T(2,3)+REF(3)*T(3,3)
C
        END
C
C
C
C
        SUBROUTINE LOOK(T,X1,X2,X3,OMEGA,TIME,IBUF,CHAN,IOSB)
C
C        DETERMINES WHERE CAMERA IS POINTING (AT)
C        FROM MMU POSITION (X1,X2,X3) AND TRANSFORMS
C        FROM RH TO LH CARTESIAN COORDINATES (FM)
C        FOR VIEWING ON EVANS & SUTHERLAND PS300.
C        DETERMINES UP VECTOR FOR PS300 AND ALSO COMPUTES
C        EARTH AND STAR ROTATIONS AND POSITION OF HORIZON.
C        INFORMATION IS THEN SENT TO THE PS300 FOR DISPLAY.
C
        INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'

        REAL CAM(3),T(3,3),X1,X2,X3
        REAL*4 OMEGA,TIME,UP(3)
        REAL*4 TROT,EROT,AT(3),FM(3),UPPS(3)
        REAL*4 NEWX(3),BMAT(3,3),BVEC(3),AMAT(3,3)
C
        INTEGER*4 CHAN,SYS$QIO,STATUS
        INTEGER*2 IOSB(4),IBUF(193)
C
C        COMPUTE MMU POSITION WRT C-W FRAME (IN LH SYSTEM)
C
        FM(1)=X1
        FM(2)=X2
        FM(3)=-X3
C
C        ASSIGN VALUES TO CAMERA LOOK VECTOR IN BODY FRAME
C
        CAM(1)=1000.
        CAM(2)=0.
        CAM(3)=0.
C
C        TRANSFORM TO REF FRAME
C
        CALL BTOR(T,CAM,AT)
C
C        ADD MMU POSITION TO CAMERA VECTOR (REF FRAME)
C        AND CONVERT TO LH SYSTEM.
C
        AT(1)=AT(1)+FM(1)
        AT(2)=AT(2)+FM(2)
        AT(3)=-AT(3)+FM(3)
C
C        ASSIGN VALUES TO UP VECTOR IN BODY FRAME
C
        UP(1)=0.
        UP(2)=0.
```

```
      UP(3)=-1000.
C
C      TRANSFORM TO REF FRAME, TRANSLATE TO 'AT' IN LH SYSTEM
C
      CALL BTOR(T,UP,UPPS)
C
      UPPS(1)=AT(1)+UPPS(1)
      UPPS(2)=AT(2)+UPPS(2)
      UPPS(3)=AT(3)-UPPS(3)
C
C      CONVERT AT,FM AND UPPS FROM FEET TO METERS
C
      AT(1)=AT(1)*.3048
      AT(2)=AT(2)*.3048
      AT(3)=AT(3)*.3048
      FM(1)=FM(1)*.3048
      FM(2)=FM(2)*.3048
      FM(3)=FM(3)*.3048
      UPPS(1)=UPPS(1)*.3048
      UPPS(2)=UPPS(2)*.3048
      UPPS(3)=UPPS(3)*.3048
C
C      DETERMINE EARTH ROTATION (DEG)
C
      EROT=TIME*.004178075
C
C      DETERMINE TARGET ROTATION (DEG)
C
      TROT=OMEGA*57.29577951*TIME
C
C      SEND INFORMATION TO PS300
C
      CALL ROT(-EROT,2,AMAT)
      CALL P919CV(AMAT,9,IBUF(169))
      CALL ROT(TROT,3,AMAT)
      CALL P919CV(AMAT,9,IBUF(125))
      CALL P919CV(AMAT,9,IBUF(147))
C
      CALL LOOKAT(AT,FM,UPPS,BMAT,BVEC)
      CALL P919CV(BMAT,9,IBUF(5))
      CALL P919CV(BVEC,3,IBUF(25))
      CALL P919CV(BMAT,9,IBUF(35))
      CALL P919CV(BVEC,3,IBUF(55))
      CALL P919CV(BMAT,9,IBUF(65))
      CALL P919CV(BVEC,3,IBUF(85))
      CALL P919CV(BMAT,9,IBUF(95))
      CALL P919CV(BVEC,3,IBUF(115))
C
      CALL NUMBER(TIME,IBUF(191))
C
C     DO A WRITE SYNC
C     42          WRITE SYNC FUNCTION CODE
C     IOSB        IO STATUS BLOCK
C     IBUF        DATA BUFFER (ACTUALLY ADDRESS OF BUFFER, BY REFERENCE)
C     386         DATA BYTE COUNT (193 WORDS)
C     0           NOT CHARACTER DATA (1 = CHARACTER DATA)
C
C     SEND ALL DATA
C
      STATUS=SYS$QIO(%VAL(1),%VAL(CHAN),%VAL(42),IOSB,,,
```

```
     +         IBUF(1),%VAL(386),%VAL(0),,,)
C
      IF (STATUS .NE. 1) THEN
        TYPE *,'BAD WRITE  <STATUS,IOSB>',STATUS,IOSB
        STOP
      ENDIF
C
      END
C
C
C
C
        SUBROUTINE ROT(ANGLE,IAXIS,AMAT)
C
C       ROUTINE TO GENERATE ROTATION REQUESTS TO PS300
C
C CALLING SEQUENCE:
C
C       CALL ROT(ANGLE,IAXIS,AMAT)
C
C WHERE:
C
C ANGLE IS THE REAL*4 ANGLE FOR ROTATION, IN DEGREES.  NEED NOT BE LIMITED
C       TO A SINGLE CIRCLE.
C IAXIS IS THE INTEGER*2 AXIS OF ROTATION (1=X, 2=Y, 3=Z).
C AMAT  IS THE REAL 3X3 MATRIX CALCULATED
C
C
        INTEGER*2 IAXIS,I,J
        REAL ANGLE,AMAT(3,3),PI180
        DATA PI180/0.017453/
C
        IDX(I)=MOD(I+2,3)+1     ! STATEMENT FUNCTION (NOT AN ARRAY)
C
        IF(IAXIS.LT.1.OR.IAXIS.GT.3) STOP 'PSLIB--AXIS OUT OF BOUNDS'
        DO 10 I=1,3
        DO 10 J=1,3
10      AMAT(I,J)=0.E0
        RADIAN = ANGLE * PI180
        C=COS(RADIAN)
        S=SIN(RADIAN)
        AMAT(IAXIS,IAXIS)=1.E0
        I=IDX(IAXIS-1)
        J=IDX(IAXIS+1)
        AMAT(I,I)=C
        AMAT(J,J)=C
        AMAT(I,J)=S
        AMAT(J,I)=-S
        RETURN
        END
C
C
C
C
      SUBROUTINE P919CV(MATRIX,N,BUFFER)
C
C       ROUTINE TO CONVERT A VAX REAL ARRAY TO ACP FLOATING-POINT FORMAT
C
C THIS ROUTINE CONVERTS AN ARRAY OF VAX SINGLE-PRECISION REAL NUMBERS INTO
C A NORMALIZED ARRAY OF 32-BIT ACP MANTISSAS, WITH THE ARRAY PRECEDED BY
```

```
C A 16-BIT EXPONENT.  THE MOST SIGNIFICANT ELEMENT IN THE ARRAY IS NORMAL-
C IZED.
C
C FORTRAN CALLING SEQUENCE:
C
C       CALL P919CV(MATRIX,N,BUFFER)
C
C WHERE:
C
C MATRIX IS AN N-ELEMENT REAL*4 ARRAY OF VAX FLOATING-POINT NUMBERS.
C N      IS THE INTEGER*2 SIZE OF ARRAY MATRIX.
C BUFFER IS THE INTEGER*2 ARRAY, OF LENGTH 2N+1, INTO WHICH THE RESULT IS
C       PLACED, WITH THE EXPONENT WORD FIRST, FOLLOWED BY THE ARRAY OF
C       32-BIT MANTISSAS.
C
C
        INTEGER*2 N,BUFFER(2*N+1),FWORD(2)
        REAL MATRIX(N),DMAX
        INTEGER*4 PSMEXP,PSMFRA,PSMNOR,DWORD,EDWORD
        EQUIVALENCE (DWORD,FWORD(1))
C       FIND LARGEST REAL NUMBER TO OBTAIN EXPONENT
        DMAX=0.
        DO 10 I=1,N
10      DMAX=AMAX1(DMAX,ABS(MATRIX(I)))
C       USE EXPONENT OF LARGEST NUMBER FOR NORMALIZATION
        DWORD=PSMEXP(DMAX)
        BUFFER(1)=FWORD(1)
        EDWORD=DWORD
C       OBTAIN NORMALIZED FRACTIONS AND LOAD INTO BUFFER
        DO 20 I=1,N
        DWORD=PSMNOR(MATRIX(I),EDWORD)
        BUFFER(2*I)=FWORD(2)
        BUFFER(2*I+1)=FWORD(1)
20      CONTINUE
        RETURN
        END
C
C
C
C
      SUBROUTINE LOOKAT(AT,FM,UP,MAT,VEC)
C
C      ROUTINE TO GENERATE LOOK AT, LOOK FROM,
C      LOOK UP REQUEST TO THE PS300
C
C      THIS ROUTINE UPDATES A PS300 DISPLAY LOOK NODE
C      WITH THE NECESSARY LOOK MATRIX
C
C      CALLING SEQUENCE:
C
C        CALL LOOKAT(AT,FM,UP,MAT,VEC)
C
C      WHERE:
C       INDEX IS THE INTEGER*2 NODE SUFFIX (1<=INDEX<=256)
C        WHICH CORRESPONDS TO ONE OF THE DISPLAY STRUCTURE
C        NODES N001 THROUGH N256.
C       AT, FM, UP : ARE THE VIEWING VECTORS
C       MAT : IS THE RESULTING 3X3 VIEWING MATRIX
C       BUFFER IS THE INTEGER*2 ARRAY, OF LENGTH 2N+1,
C        INTO WHICH THE RESULT IS PLACED, WITH THE EXPONENT
```

```
C          WORD FIRST, FOLLOWED BY THE ARRAY OF 32-BIT MANTISSAS.
C         VEC IS A 3 ELEMENT ARRAY CONTAINING ROW 4
C
C       COMPUTES 4X4 MATRIX FOR LOOK FUNCTION
C
      REAL AT(3),FM(3),UP(3),MAT(3,3),T(3),VEC(3)
      REAL D(3),E(3),M,F(3),G(3),H(3),MAG
C
      D(1)=AT(1)-FM(1)
      D(2)=AT(2)-FM(2)
      D(3)=AT(3)-FM(3)
C
      MAG=D(1)**2+D(2)**2+D(3)**2
      IF (MAG .GT. .1E-30) THEN
        MAG=SQRT(MAG)
       ELSE
        MAG=.1E-30
      ENDIF
C
      D(1)=D(1)/MAG
      D(2)=D(2)/MAG
      D(3)=D(3)/MAG
C
      E(1)=UP(1)-AT(1)
      E(2)=UP(2)-AT(2)
      E(3)=UP(3)-AT(3)
C
      M=D(1)*E(1)+D(2)*E(2)+D(3)*E(3)
C
      F(1)=E(1)-M*D(1)
      F(2)=E(2)-M*D(2)
      F(3)=E(3)-M*D(3)
C
      IF (((F(1)+F(2)+F(3)) .EQ. 0.0) THEN
        E(1)=0.0
        E(2)=1.0
        E(3)=0.0
        F(1)=E(1)-M*D(1)
        F(2)=E(2)-M*D(2)
        F(3)=E(3)-M*D(3)
        IF (((F(1)+F(2)+F(3)) .EQ. 0.0) THEN
          E(1)=0.0
          E(2)=0.0
          E(3)=1.0
          F(1)=E(1)-M*D(1)
          F(2)=E(2)-M*D(2)
          F(3)=E(3)-M*D(3)
        ENDIF
      ENDIF
C
      MAG=SQRT(F(1)**2+F(2)**2+F(3)**2)
C
      G(1)=F(1)/MAG
      G(2)=F(2)/MAG
      G(3)=F(3)/MAG
C
      H(1)=G(2)*D(3)-G(3)*D(2)
      H(2)=G(3)*D(1)-G(1)*D(3)
      H(3)=G(1)*D(2)-G(2)*D(1)
C
```

```
      T(1)=-FM(1)*H(1)-FM(2)*H(2)-FM(3)*H(3)
      T(2)=-FM(1)*G(1)-FM(2)*G(2)-FM(3)*G(3)
      T(3)=-FM(1)*D(1)-FM(2)*D(2)-FM(3)*D(3)
C
      MAT(1,1)=H(1)
      MAT(1,2)=H(2)
      MAT(1,3)=H(3)
      VEC(1)=T(1)
C
      MAT(2,1)=G(1)
      MAT(2,2)=G(2)
      MAT(2,3)=G(3)
      VEC(2)=T(2)
C
      MAT(3,1)=D(1)
      MAT(3,2)=D(2)
      MAT(3,3)=D(3)
      VEC(3)=T(3)
C
C
      RETURN
      END
C
C
C
C
      SUBROUTINE P920CV(MAT,VEC,BUFFER)
C
C     THIS SUBROUTINE FILLS IN A 4 X 4 ARRAY INTO THE BUFFER.
C     YOU LOAD A 3 X 4 MAT, AND AN ARRAY VEC(3) IS THE FOURTH ROW.
C     IT FILLS 34 ELEMENTS IN THE BUFFER.
C
      INTEGER*2 ITEMP1(9), ITEMP2(25), BUFFER(34)
      REAL MAT(3,4), VEC(4),MAT1(4,3)
      DO 100 I=1,3
      DO 99  J=1,4
        MAT1(J,I)=MAT(I,J)
 99   CONTINUE
 100  CONTINUE
      CALL P919CV(VEC,4,ITEMP1)
      CALL P919CV(MAT1,12,ITEMP2)
      BUFFER(1)  =  ITEMP1(1)
      DO 5 I = 1,25
         BUFFER(I+1) = ITEMP2(I)
  5   CONTINUE
      DO 10 J = 1,8
         BUFFER(J+26) = ITEMP1(J+1)
  10  CONTINUE
      RETURN
      END


      SUBROUTINE NUMBER(X,BUFFER)
C
C     THIS SUBROUTINE WILL TAKE ANY NUMBER FROM 9999.9 TO -999.9,
C     ENCODE IT, AND PUT IT IN THE PROPER PLACE IN THE OUTPUT
C     BUFFER, IBUF

      INTEGER*2 BUFFER(3), TBUF(3)
      CHARACTER*6 CHAR
```

```
      LOGICAL *1 TEMP0,TEMP1(6)
      EQUIVALENCE (CHAR,TEMP1,TBUF)

      IF(X.GT.9999.9) X=9999.9
      IF(X.LT.-999.9) X=-999.9
         ENCODE(6,201,CHAR)X
201     FORMAT(F6.1)

         TEMP0=TEMP1(1)
         TEMP1(1)=TEMP1(2)
         TEMP1(2)=TEMP0
         TEMP0=TEMP1(3)
         TEMP1(3)=TEMP1(4)
         TEMP1(4)=TEMP0
         TEMP0=TEMP1(5)
         TEMP1(5)=TEMP1(6)
         TEMP1(6)=TEMP0
         BUFFER(1)=TBUF(1)
         BUFFER(2)=TBUF(2)
         BUFFER(3)=TBUF(3)
         RETURN
         END
C
C
C
C
      SUBROUTINE INITBUF(IBUF,CHAN,IOSB)
C
C       THIS SUBROUTINE FILLS IBUF WITH DATA THAT DOES NOT CHANGE
C       DURING PROGRAM.  THIS DATA IS USED TO ADDRESS PS300 MEMORY
C       LOCATIONS.
C
        INTEGER*4 SYS$QIO,SYS$SETEF,SYS$WAITFR
        INTEGER*4 CHAN,STATUS,SYS$ASSIGN,SYS$QIOW,NAMADX(8)
        INTEGER*2 IBUF(193),NAMES(5,8),NAMADR(2,8),IOSB(4),BUFNUM
        CHARACTER*4 UNIT
        EQUIVALENCE (NAMADR,NAMADX)
        DATA NAMES/'TA','RG','T.','LO','OK','PL','BA','Y.','LO','OK',
     1             'GL','OB','E.','LO','OK','ST','AR','S.','LO','OK',
     1             'GL','OB','E.','TR','OT','ST','AR','S.','TR','OT',
     1             'GL','OB','E.','ER','OT','IN','FO','-R','TI','ME'/

      DATA UNIT/'PIA0'/
C
C       WAIT
C
      DO 5 J=1,500000
5     CONTINUE
C
C       SET UP HOUSEKEEPING
C
C       GET A CHANNEL NUMBER
C
        STATUS=SYS$ASSIGN(UNIT,CHAN,,)
        IF(STATUS.NE.1) THEN
          TYPE *,'BAD ASSIGN!  <STATUS> = ',STATUS
          STOP
        ENDIF
C
C       DETACH FOR SAFETY:      34 --> DETACH FUNCTION CODE
```

```
C
10      STATUS=SYS$QIOW(,%VAL(CHAN),%VAL(34),IOSB,,,,,,,,)
C
C       ATTACH:                    33 --> ATTACH FUNCTION CODE
C
        STATUS=SYS$QIOW(,%VAL(CHAN),%VAL(33),IOSB,,,,,,,,)
        IF(STATUS.NE.1) THEN
          TYPE *,'BAD ATTACH!  <STATUS> = ',STATUS
          STOP
        ENDIF
C
C       GET THE ADDRESSES OF THE ENTITIES TO UPDATE
C       43 --> LOOKUP NAMED ENTITIES FUNCTION CODE
C
20      DO 25 I=1,8
          STATUS=SYS$QIOW(,%VAL(CHAN),%VAL(43),IOSB,,,NAMES(1,I),
     1            %VAL(10),%VAL(1),,,)
C
          IF(STATUS.EQ.1.AND.IOSB(1).EQ.1.AND.
     1         (IOSB(3).OR.IOSB(4)).NE.0) GOTO 21
          TYPE *,'BAD ENTITY FETCH! <STAT,IOSB> -- ',STATUS,IOSB
          STOP
C
C         GET THE ADDRESS FROM OUT OF THE IO STATUS BLOCK (IOSB)
C
21        DO 24 J=1,2
            NAMADR(J,I)=IOSB(J+2)
24        CONTINUE
25      CONTINUE
C
C
C       OFFSET THE ADDRESSES TO GET PAST THE FIRST THREE FIELDS
C       OF THE NODE WHICH WE DO NOT WANT TO CHANGE.
C
        DO 30 I=1,8
           NAMADX(I)=NAMADX(I)+8
30      CONTINUE
C
C       OFFSET TEXT BY AN ADDITIONAL 16
C
        NAMADX(8)=NAMADX(8)+16
C
C
C       BUFFER 1  SETUP
C       TRANSLATION NEEDS  7 ELEMENTS
C       ROTATION NEEDS    19 ELEMENTS
C       LOOKAT NEEDS      28 ELEMENTS
C       TEXT NEEDS         3 ELEMENTS
C
C
        IBUF(1)          = 8              ! FIVE BLOCKS
        IBUF(2)          = NAMADR(1,1)    ! BLOCK ONE ADDRESS - TARGT.LOOK
        IBUF(3)          = NAMADR(2,1)    ! BLOCK ONE ADDRESS
        IBUF(4)          = 27             ! WORD COUNT FOR BLOCK 1
        IBUF(24)         = 1              ! TRAN FLAG
        IBUF(32)         = NAMADR(1,2)    ! BLOCK TWO ADDRESS - PLBAY.LO
        IBUF(33)         = NAMADR(2,2)    ! BLOCK TWO ADDRESS
        IBUF(34)         = 27             ! WORD COUNT FOR BLOCK 2
        IBUF(54)         = 1              ! TRAN FLAG
        IBUF(62)         = NAMADR(1,3)    ! BLOCK 3 ADDRESS - GLOBE.LOCK
```

```
      IBUF(63)        = NAMADR(2,3)   ! BLOCK 3 ADDRESS
      IBUF(64)        = 27            ! WORD COUNT FOR BLOCK 3
      IBUF(84)        = 1             ! TRAN FLAG
      IBUF(92)        = NAMADR(1,4)   ! BLOCK 4 ADDRESS - STARS.LOOK
      IBUF(93)        = NAMADR(2,4)   ! BLOCK 4 ADDRESS
      IBUF(94)        = 27            ! WORD COUNT FOR BLOCK 4
      IBUF(114)       = 1             ! TRAN FLAG

      IBUF(122)       = NAMADR(1,5)   ! BLOCK 5 ADDRESS - GLOBE.TROT
      IBUF(123)       = NAMADR(2,5)   ! BLOCK 5 ADDRESS
      IBUF(124)       = 19            ! WORD COUNT FOR BLOCK 5
      IBUF(144)       = NAMADR(1,6)   ! BLOCK 6 ADDRESS - STARS.TROT
      IBUF(145)       = NAMADR(2,6)   ! BLOCK 6 ADDRESS
      IBUF(146)       = 19            ! WORD COUNT FOR BLOCK 6


      IBUF(166)       = NAMADR(1,7)   ! BLOCK 7 ADDRESS - GLOBE.EROT
      IBUF(167)       = NAMADR(2,7)   ! BLOCK 7 ADDRESS
      IBUF(168)       = 19            ! WORD COUNT FOR BLOCK 7

      IBUF(188)       = NAMADR(1,8)   ! BLOCK 8 ADDRESS - ONFO.RTIME
      IBUF(189)       = NAMADR(2,7)   ! BLOCK 8 ADDRESS
      IBUF(190)       = 3             ! WORD COUNT FOR BLOCK 8
C
      END
C
C
C
C

      SUBROUTINE ROTATE(ANG,DELTAT,WB,Q,T)
C
C     LINEARLY INTEGRATES TO FIND ROTATION IN BODY FRAME, THEN
C     TRANSFORMS TO REFERENCE FRAME.
C
      REAL ANG(3),WB(3),DELTAT,Q(4),T(3,3)
C
C       COMPUTE BODY RATES (WB)
C       (ANG IS ANGULAR ACCELERATION)
C
      WB(1)=WB(1)+DELTAT*ANG(1)
      WB(2)=WB(2)+DELTAT*ANG(2)
      WB(3)=WB(3)+DELTAT*ANG(3)
C
C       FIND QUATERNION RATE AND INTEGRATE
C
      CALL QDOT(Q,WB(1),WB(2),WB(3),DELTAT)
C
C       COMPUTE ROTATION MATRIX
C
      CALL TRNSFM(T,Q)
C
      END
C
C
C
C

      SUBROUTINE PS300(ALT,INCLIN)
C
C     THIS SUBROUTINE IS THE GRAPHICS PROGRAM FOR THE
```

```
C       MMU SIMULATOR. IT GENERATES A TARGET, ROTATING
C       EARTH, HORIZON AND STAR FIELD FOR BACKGROUND.
C       SUBROUTINE LOOK SENDS NEW DATA TO THIS PROGRAM IN
C       THE PS300 TO UPDATE TARGET RANGE AND ATTITUDE AND
C       EARTH/STAR ROTATIONS.
C
        INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'
C
        LOGICAL*1 POSLIN(73)
        REAL*4 ALT,INCLIN,AT(3),FM(3),UP(3)
        REAL*4 V(3),HORR,HORT,HORD,LONG,ANG
        INTEGER I,NVEC
        DIMENSION VECS(4,73)
C
        DATA POSLIN/.FALSE.,72*.TRUE./
C
C         ATTACH GRAPHICS DEVICE AND INITIALIZE GRAPHICS
C
        CALL PATTCH('LOGDEVNAM=PIA0:/PHYDEVTYP=PARALLEL',ERR)
        CALL PINIT(ERR)
C
C         V(3) - VECTOR ARRAY FOR PS300
C         HORR - RADIUS OF EARTH HORIZON AS SEEN FROM TARGET (M)
C         HORD - DISTANCE OF HORIZON FROM EARTH CENTER (M)
C         HORT - DISTANCE OF HORIZON FROM TARGET (M)
C
        CALL BAY
C
C         COMPUTE EARTH HORIZON RADIUS (HORR),
C         DISTANCE FROM EARTH CENTER (HORD)(M),
C         AND DISTANCE FROM TARGET TO HORIZON (HORT)
C
        ALT=ABS(ALT)
        HORD=40680606.08/(6378.135+ALT)
        HORR=SQRT(40680606.08-HORD*HORD)
        HORT=HORR*6378.135/HORD
        HORD=HORD*1000.0
        HORR=HORR*1000.0
        HORT=HORT*1000.0
C
C         INITIALIZE PS300 INSTANCE NODES AND CONNECTIONS
C
        V(1)=0.0
        V(2)=0.0
        V(3)=0.0
C
        AT(1)=0.0
        AT(2)=0.0
        AT(3)=20.0
C
        FM(1)=0.0
        FM(2)=0.0
        FM(3)=0.0
C
        UP(1)=0.0
        UP(2)=10.0
        UP(3)=0.0
C
        CALL PBEGS('INFO',ERR)
          CALL PSEDCL('CLIP',.FALSE.,'"',ERR)
```

```
      CALL PVIEWP('',-1.,1.,-1.,1.,1.,1.,'',ERR)
      CALL PWINDO('WNDW',0.,80.,0.,80.,0.,100.,'',ERR)
      CALL PLOOKA('LOOK',AT,FM,UP,'',ERR)
      CALL PSECHW('','',ERR)
      CALL PCHSCA('',1.,1.,'',ERR)
      CALL PCHS('',0.,79.,1.,1.,0.,'USAFA ASTRO LAB',ERR)
      CALL PCHS('',0.,75.,1.,1.,0.,'TIME (SEC)',ERR)
      CALL PCHS('RTIME',12.,75.,1.,1.,0.,'000000',ERR)
    CALL PENDS(ERR)
C
    CALL PBEGS('TARGT',ERR)
C   UNITS IN METERS
      CALL PSEDCL('CLIP',.TRUE.,'',ERR)
      CALL PVIEWP('',-1.0,1.0,-1.0,1.0,1.0,1.0,'',ERR)
      CALL PFOV('',35.0,1.0,HORT,'',ERR)
      CALL PLOCKA('LOOK',AT,FM,UP,'',ERR)
      CALL PINST('','SHUTTLE',ERR)
    CALL PENDS(ERR)
C
C       BUILD A SPACE SHUTTLE
C
    CALL VECTOR('RWING',5)
    CALL PBEGS('SHUTTLE',ERR)
         V(1)=1.395
         V(2)=1.395
         V(3)=1.395
      CALL PSCALE('',V,'',ERR)
      CALL PROTX('',-90.0,'',ERR)
      CALL PROTY('',-2.0,'',ERR)
         V(1)=-19.
         V(2)=0.0
         V(3)=-2.35
      CALL PTRANS('',V,'',ERR)
      CALL PINST('','DOOR',ERR)
      CALL PINST('','RWING',ERR)
      CALL PINST('','RMAIN',ERR)
      CALL PINST('','CMAIN',ERR)
         V(1)=1.0
         V(2)=-1.0
         V(3)=1.0
      CALL PSCALE('',V,'DOOR',ERR)
      CALL PSCALE('',V,'RWING',ERR)
      CALL PSCALE('',V,'RMAIN',ERR)
    CALL PENDS(ERR)
C
C       BUILD THE RIGHT MAIN ENGINE
C
    CALL PBEGS('RMAIN',ERR)
        V(1)=2.8
        V(2)=1.0
        V(3)=1.5
      CALL PTRANS('',V,'',ERR)
      CALL PROTY('',10.0,'',ERR)
      CALL PROTZ('',-5.0,'ENGINE',ERR)
    CALL PENDS(ERR)
C
C       BUILD A CENTER ENGINE
C
    CALL PBEGS('CMAIN',ERR)
        V(1)=3.2
```

```
          V(2)=0.0
          V(3)=3.6
         CALL PTRANS('*',V,'*',ERR)
         CALL PROTY('*',15.0,'ENGINE',ERR)
      CALL PENDS(ERR)
C
C       BUILD A MAIN ENGINE
C
      CALL PBEGS('ENGINE',ERR)
        CALL PSECOL('*',120.0,.8,'*',ERR)
        CALL PROTZ('*',-90.0,'*',ERR)
          V(1)=0.0
          V(2)=-2.5
          V(3)=0.0
        CALL PTRANS('*',V,'*',ERR)
          V(1)=1.0
          V(2)=2.5
          V(3)=1.0
        CALL PSCALE('*',V,'*',ERR)
        CALL PINST('*','HEMI',ERR)
        CALL PROTX('*',-90.0,'SEMI',ERR)
        CALL PROTX('*',90.0,'SEMI',ERR)
      CALL PENDS(ERR)
C
C       BUILD A MOVABLE DOOR
C
      CALL VECTOR('RDOOR',5)
      CALL PBEGS('DOOR',ERR)
          V(1)=19.0
          V(2)=1.9
          V(3)=2.35
        CALL PTRANS('*',V,'*',ERR)
        CALL PROTY('*',2.0,'*',ERR)
        CALL PROTX('OPEN',-140.0,'*',ERR)
        CALL PROTY('*',-2.0,'*',ERR)
          V(1)=-19.0
          V(2)=-1.9
          V(3)=-2.35
        CALL PTRANS('*',V,'RDOOR',ERR)
      CALL PENDS(ERR)
C
C       COMMANDS FOR DOOR OPENING
C
C       SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION
C
      CALL PFN('DOORROT','XROTATE',ERR)
      CALL PSNST('DOOR',1,'DLABEL8',ERR)
C
C       CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC8','ACCUMULATE',ERR)
C
      CALL PCONN('DIALS',8,1,'ACC8',ERR)
      CALL PSNREA(-140.,2,'ACC8',ERR)
      CALL PSNREA(1.,3,'ACC8',ERR)
      CALL PSNREA(-15.,4,'ACC8',ERR)
      CALL PSNREA(0.,5,'ACC8',ERR)
      CALL PSNREA(-140.,6,'ACC8',ERR)
C
      CALL PCONN('ACC8',1,1,'DOORROT',ERR)
```

```
      CALL PCONN('DOORROT',1,1,'DOOR.OPEN',ERR)
C
C
C       CREATE A SPINNING/INCLINED GLOBE OF THE EARTH,
C       SET IT COUNTER-ROTATING W.R.T. TARGET ORBIT.
C       ADD A STATIONARY HORIZON
C
      CALL PBEGS('GLOBE',ERR)
        CALL PSEDCL('CLIP',.TRUE.,'',ERR)
        CALL PVIEWP('',-1.0,1.0,-1.0,1.0,1.0,1.0,'',ERR)
        CALL PFOV('',35.0,ALT*100.0,HORT+200000.0,'',ERR)
        CALL PLOOKA('LOOK',AT,FM,UP,'',ERR)
          V(1)=0.0
          V(2)=(-6378.135-ALT)*1000.0
          V(3)=0.0
        CALL PTRANS('',V,'',ERR)
        CALL PINST('','HORIZON',ERR)
        CALL PROTZ('TROT',0.0,'',ERR)
        CALL PSECOL('',240.,1.,'',ERR)
        CALL PROTX('',90.-INCLIN,'',ERR)
        CALL PROTY('EROT',0.0,'',ERR)
          V(1)=6378135.
          V(2)=V(1)
          V(3)=V(1)
        CALL PSCALE('',V,'',ERR)
        CALL PINST('','WORLD',ERR)
        CALL PSECOL('',240.,0.0,'',ERR)
        CALL PINST('','SPHERE',ERR)
        CALL PINST('','LATLINE',ERR)
      CALL PENDS(ERR)
C
C       CREATE A HORIZON FROM A CIRCLE
C
      CALL PBEGS('HORIZON',ERR)
          V(1)=0.0
          V(2)=HORD
          V(3)=0.0
        CALL PTRANS('',V,'',ERR)
        CALL PROTX('',90.,'',ERR)
          V(1)=HORR
          V(2)=HORR
          V(3)=HORR
        CALL PSCALE('',V,'CIRCLE',ERR)
      CALL PENDS(ERR)
C
C       CREATE A STAR FIELD AND SET IT
C       COUNTER-ROTATING W.R.T. TARGET ORBIT
C
      CALL PBEGS('STARS',ERR)
C     UNITS ARE IN METERS
        CALL PSEDCL('CLIP',.TRUE.,'',ERR)
        CALL PVIEWP('',-1.0,1.0,-1.0,1.0,1.0,1.0,'',ERR)
          HORT=HORT+900.*SQRT((2.*ALT+6378.)**2-6378.**2)
        CALL PFOV('',35.0,.1*HORT,HORT,'',ERR)
        CALL PLOOKA('LOOK',AT,FM,UP,'',ERR)
          V(1)=0.0
          V(2)=-6378135.0-ALT*1000.0
          V(3)=0.0
        CALL PTRANS('',V,'',ERR)
        CALL PROTZ('TROT',0.0,'',ERR)
```

```
        CALL PSECOL('*',180.,0.,'*',ERR)
        CALL PROTZ('*',180.,'STARS.TWINKLE',ERR)
        CALL PROTY('*',180.,'STARS.TWINKLE',ERR)
          V(1)=6378135.0+ALT*2000.0
          V(2)=V(1)
          V(3)=V(1)
        CALL PSCALE('TWINKLE',V,'STAR',ERR)
      CALL PENDS(ERR)
C
C       BUILD A SPHERE FROM A CIRCLE
C
      CALL PBEGS('SPHERE',ERR)
        DO 200 LONG=10,180,10
          CALL PROTY('*',LONG,'CIRCLE',ERR)
200     CONTINUE
      CALL PENDS(ERR)
C
C
C       BUILD A HEMI-SPHERE FROM A SEMI-CIRCLE
C
      CALL PBEGS('HEMI',ERR)
        DO 222 LONG=15,180,15
          CALL PROTY('*',LONG,'SEMI',ERR)
222     CONTINUE
      CALL PENDS(ERR)
C
C
C       COMPUTE LINES OF LATITUDE USING CIRCLES
C
      CALL PBEGS('LATLINE',ERR)
        CALL PROTX('*',90.0,'CIRCLE',ERR)
        CALL PROTX('*',90.0,'LAT',ERR)
        CALL PROTX('*',-90.0,'LAT',ERR)
      CALL PENDS(ERR)
C
C
      CALL PBEGS('LAT',ERR)
        DO 20 I=10,80,10
           ANG=I*.0174532925
           V(1)=0.0
           V(2)=0.0
           V(3)=SIN(ANG)-SIN(ANG-.174532925)
         CALL PTRANS('*',V,'*',ERR)
           V(1)=COS(ANG)
           V(2)=V(1)
           V(3)=0.0
         CALL PSCALE('*',V,'CIRCLE',ERR)
20      CONTINUE
      CALL PENDS(ERR)
C
C
C       VECTOR LIST FOR CIRCLE
C
      DO 10 I=1,73
        ANG=5.0*(I-1)*.0174532925
        VECS(1,I)=COS(ANG)
        VECS(2,I)=SIN(ANG)
        VECS(3,I)=0.0
        VECS(4,I)=1.0
10    CONTINUE
```

```
C
      NVEC=73
      CALL PVCBEG('CIRCLE',NVEC,.TRUE.,.FALSE.,3,PVITEM,ERR)
        CALL PVCLIS(NVEC,VECS,POSLIN,ERR)
      CALL PVCEND(ERR)
C
C
C       VECTOR LIST FOR SEMI-CIRCLE
C
      DO 12 I=1,13
        ANG=(-90.0+15.0*(I-1))*.0174532925
        VECS(1,I)=SIN(ANG)
        VECS(2,I)=COS(ANG)
        VECS(3,I)=0.0
        VECS(4,I)=1.0
12    CONTINUE
C
      NVEC=13
      CALL PVCBEG('SEMI',NVEC,.TRUE.,.FALSE.,3,PVITEM,ERR)
        CALL PVCLIS(NVEC,VECS,POSLIN,ERR)
      CALL PVCEND(ERR)
C
C
C       CALL NEEDED VECTOR LISTS
C
      CALL VECTOR ('WORLD',5)
      CALL VECTOR ('STAR',4)
C
C       DISPLAY ALL
C
      CALL PDISP('INFO',ERR)
      CALL PDISP('TARGT',ERR)
      CALL PDISP('GLOBE',ERR)
      CALL PDISP('PLBAY',ERR)
      CALL PDISP('STARS',ERR)
C
C
C         DETACH GRAPHICS
C
      CALL PDTACH(ERR)
C
      END
C
C
C
      SUBROUTINE VECTOR (NAME,LENGTH)
C
C       THIS SUBROUTINE READS A VECTOR LIST FROM VAX
C       FILE AND PUTS IT IN USABLE FORM FOR THE PS300
C
      INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'

      INTEGER*4 IPOS, LENGTH, CLASS
      REAL*4 POINTS (4,2000)
      LOGICAL*1 POSLIN (2000), PL, VL
      CHARACTER NAME*8, FILENAME*26

      FILENAME='[ALFANO.DATA]'//NAME(:LENGTH)//'.DAT'
      OPEN ( UNIT=1, NAME=FILENAME,TYPE='OLD',READONLY)
      READ ( 1, 910) VL,IPOS
```

```
910     FORMAT ( A1,I10)
        IF ((VL.EQ.'C') .OR. (VL.EQ.'C'))  THEN
          CLASS=0
        ELSE IF ((VL.EQ.'D') .OR. (VL .EQ.'D')) THEN
          CLASS=1
        ELSE IF ((VL.EQ.'I') .OR. (VL .EQ.'I')) THEN
          CLASS=2
        ELSE IF ((VL.EQ.'S') .OR. (VL .EQ.'S')) THEN
          CLASS=3
        ENDIF
        DO 3410 I=1,IPOS
        READ ( 1, 911) PL,(POINTS(K,I),K=1,3)
911     FORMAT ( A1, 3F12.8)
        POINTS (4,I)=1
        POSLIN(I)=.FALSE.
        IF ((PL.EQ.'L') .OR.( PL.EQ.'L')) POSLIN(I)=.TRUE.
3410    CONTINUE
        CALL PVCBEG (NAME(:LENGTH),IPOS,.FALSE.,.FALSE.,3,CLASS,ERR)
         CALL PVCLIS (IPOS,POINTS,POSLIN,ERR)
        CALL PVCEND (ERR)
        CLOSE (UNIT=1)
        END
C
C
C
C
        SUBROUTINE UNITIZE(V)
C
C       UNITIZES V VECTOR
C
        REAL V(3),MAGV
C
        IF (ABS(V(1)) .LT. .1E-10) V(1)=.1E-10
        IF (ABS(V(2)) .LT. .1E-10) V(2)=.1E-10
        IF (ABS(V(3)) .LT. .1E-10) V(3)=.1E-10
        MAGV=SQRT(V(1)*V(1)+V(2)*V(2)+V(3)*V(3))
        V(1)=V(1)/MAGV
        V(2)=V(2)/MAGV
        V(3)=V(3)/MAGV
C
        END
C
C
C
C
        SUBROUTINE CROSS(A,B,C)
C
C       COMPUTES  A = B X C
C
        REAL A(3),B(3),C(3)
C
        A(1)=B(2)*C(3)-B(3)*C(2)
        A(2)=B(3)*C(1)-B(1)*C(3)
        A(3)=B(1)*C(2)-B(2)*C(1)
C
        END
C
C
C
C
```

```
      SUBROUTINE QDOT(Q,WX,WY,WZ,DELTAT)
C
C     FIGURES QUATERNION RATE (QD) AND LINEARLY INTEGRATES
C     FOR TIME STEP DELTAT.
C
      REAL Q(4),QD(4),WX,WY,WZ,DELTAT,CONST
      INTEGER J
C
      QD(1)=(Q(2)*WZ-Q(3)*WY+Q(4)*WX)*.5
      QD(2)=(-Q(1)*WZ+Q(3)*WX+Q(4)*WY)*.5
      QD(3)=(Q(1)*WY-Q(2)*WX+Q(4)*WZ)*.5
      QD(4)=(-Q(1)*WX-Q(2)*WY-Q(3)*WZ)*.5
C
      DO 100 J=1,4
        Q(J)=Q(J)+DELTAT*QD(J)
100   CONTINUE
C
C       NORMALIZE QUATERNIONS
C
      CONST=SQRT(Q(1)*Q(1)+Q(2)*Q(2)+Q(3)*Q(3)+Q(4)*Q(4))
      Q(1)=Q(1)/CONST
      Q(2)=Q(2)/CONST
      Q(3)=Q(3)/CONST
      Q(4)=Q(4)/CONST
C
      END
C
C
C
C
      SUBROUTINE TRNSFM(T,Q)
C
C     COMPUTES TRANSFORMATION MATRIX (T) FROM QUATERNIONS (Q)
C
      REAL T(3,3),Q(4)
C
      T(1,1)=Q(1)*Q(1)-Q(2)*Q(2)-Q(3)*Q(3)+Q(4)*Q(4)
      T(2,1)=2.0*(Q(1)*Q(2)+Q(3)*Q(4))
      T(3,1)=2.0*(Q(1)*Q(3)-Q(2)*Q(4))
      T(1,2)=2.0*(Q(1)*Q(2)-Q(3)*Q(4))
      T(2,2)=-Q(1)*Q(1)+Q(2)*Q(2)-Q(3)*Q(3)+Q(4)*Q(4)
      T(3,2)=2.0*(Q(2)*Q(3)+Q(1)*Q(4))
      T(1,3)=2.0*(Q(1)*Q(3)+Q(2)*Q(4))
      T(2,3)=2.0*(Q(2)*Q(3)-Q(1)*Q(4))
      T(3,3)=-Q(1)*Q(1)-Q(2)*Q(2)+Q(3)*Q(3)+Q(4)*Q(4)
C
      END
C
C
C
C
        SUBROUTINE IOBUFF (IFUNC,C,RESET)

        INTEGER S(24),L(24),C(12)
        INTEGER IFUNC,RESET
        INTEGER*2 DATAOUT,DATAIN
        INTEGER*2 DIBUF(10),DOBUF(10)
        INTEGER*4 ISTAT,ICALL
        INTEGER*4 NCHAN,DOFLAG,DIFLAG
```

```
      REAL ARATE,RATE

      DATA DOFLAG,DIFLAG/3,5/
      DATA NFRAME/10/
      DATA MODEOUT,MODEIN/8,7/
      DATA IUNIT/1/
      DATA ICHAN/1/
      DATA NCHAN/1/
      DATA ISMODE/0/
      DATA INIT/1/
      DATA RATE/80000.0/

      GO TO (100,200) IFUNC

100   CALL LPAIO (INIT,IUNIT,,RATE,,,,,ARATE,ISTAT,ICALL,,)
      IF (.NOT. ISTAT) GO TO 950
      RETURN

200   CONTINUE
      DATAOUT='0000'X
      DO 210 I=1,6
210   DATAOUT=DATAOUT+L(I)*2**(I-1)
      DO 220 I=1,NFRAME
220   DOBUF(I)=DATAOUT .XOR. 'FFFF'X
      CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DOFLAG))
230   CALL LPAIO (18,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 230
      CALL LPAIO (MODEIN,IUNIT,DIFLAG,,ICHAN,NCHAN,NFRAME,DIBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DIFLAG))
240   CALL LPAIO (17,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 240
      DATAIN=(DIBUF(1).XOR.'FFFF'X)
      S(1)=DATAIN .AND. 1
      S(2)=(DATAIN .AND. 2)/2
      S(3)=(DATAIN .AND. 4)/4
      S(4)=(DATAIN .AND. 8)/8
      S(5)=(DATAIN .AND. 16)/16
      S(6)=(DATAIN .AND. 32)/32
      RESET=((64.XOR.DATAIN) .AND. 64)/64
      DATAOUT='4000'X
      DO 250 I=7,15
250   DATAOUT=DATAOUT+L(I)*2**(I-7)
      DO 260 I=1,NFRAME
260   DOBUF(I)=DATAOUT .XOR. 'FFFF'X
      CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DOFLAG))
270   CALL LPAIO (18,,,,,,,,,ISTAT,ICALL,LSTAT,)
      IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 270
      CALL LPAIO (MODEIN,IUNIT,DIFLAG,,ICHAN,NCHAN,NFRAME,DIBUF,,
     1 ISTAT,ICALL,,ISMODE)
      IF (.NOT. ISTAT) GO TO 950
      CALL SYS$WAITFR (%VAL(DIFLAG))
```

```
280     CALL LPAIO (17,,,,,,,,,ISTAT,ICALL,LSTAT,)
        IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 280
        DATAIN=(DIBUF(1).XOR.'FFFF'X)
        S(7)=DATAIN .AND. 1
        S(8)=(DATAIN .AND. 2)/2
        S(9)=(DATAIN .AND. 4)/4
        S(10)=(DATAIN .AND. 8)/8
        S(11)=(DATAIN .AND. 16)/16
        S(12)=(DATAIN .AND. 32)/32
        S(13)=(DATAIN .AND. 64)/64
        S(14)=(DATAIN .AND. 128)/128
        S(15)=(DATAIN .AND. 256)/256
        DATAOUT='8000'X
        DO 290 I=16,24
290     DATAOUT=DATAOUT+L(I)*2**(I-16)
        DO 300 I=1,NFRAME
300     DOBUF(I)=DATAOUT .XOR. 'FFFF'X
        CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
     1 ISTAT,ICALL,,ISMODE)
        IF (.NOT. ISTAT) GO TO 950
        CALL SYS$WAITFR (%VAL(DOFLAG))
310     CALL LPAIO (18,,,,,,,,,ISTAT,ICALL,LSTAT,)
        IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 310
        CALL LPAIO (MODEIN,IUNIT,DIFLAG,,ICHAN,NCHAN,NFRAME,DIBUF,,
     1 ISTAT,ICALL,,ISMODE)
        IF (.NOT. ISTAT) GO TO 950
        CALL SYS$WAITFR (%VAL(DIFLAG))
320     CALL LPAIO (17,,,,,,,,,ISTAT,ICALL,LSTAT,)
        IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 320
        DATAIN=(DIBUF(1).XOR.'FFFF'X)
        S(16)=DATAIN .AND. 1
        S(17)=(DATAIN .AND. 2)/2
        S(18)=(DATAIN .AND. 4)/4
        S(19)=(DATAIN .AND. 8)/8
        S(20)=(DATAIN .AND. 16)/16
        S(21)=(DATAIN .AND. 32)/32
        S(22)=(DATAIN .AND. 64)/64
        S(23)=(DATAIN .AND. 128)/128
        S(24)=(DATAIN .AND. 256)/256
        DATAOUT='C000'X
        DO 340 I=1,NFRAME
340     DOBUF(I)=DATAOUT .XOR. 'FFFF'X
        CALL LPAIO (MODEOUT,IUNIT,DOFLAG,,ICHAN,NCHAN,NFRAME,DOBUF,,
     1 ISTAT,ICALL,,ISMODE)
        IF (.NOT. ISTAT) GO TO 950
        CALL SYS$WAITFR (%VAL(DOFLAG))
350     CALL LPAIO (18,,,,,,,,,ISTAT,ICALL,LSTAT,)
        IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 350
        CALL LPAIO (MODEIN,IUNIT,DIFLAG,,ICHAN,NCHAN,NFRAME,DIBUF,,
     1 ISTAT,ICALL,,ISMODE)
        IF (.NOT. ISTAT) GO TO 950
        CALL SYS$WAITFR (%VAL(DIFLAG))
360     CALL LPAIO (17,,,,,,,,,ISTAT,ICALL,LSTAT,)
        IF ((ISTAT.NE.-1).AND.(LSTAT.NE.1)) GO TO 360
        DATAIN=(DIBUF(1).XOR.'FFFF'X)
        C(1)=(1.XOR.DATAIN) .AND. 1
        C(2)=((2.XOR.DATAIN) .AND. 2)/2
        C(3)=(DATAIN .AND. 4)/4
        C(4)=(DATAIN .AND. 8)/8
        C(5)=(DATAIN .AND. 16)/16
```

```
          C(6)=(DATAIN .AND. 32)/32
          C(7)=(DATAIN .AND. 64)/64
          C(8)=(DATAIN .AND. 128)/128
          C(9)=(DATAIN .AND. 256)/256
          C(10)=(DATAIN .AND. 512)/512
          C(11)=(DATAIN .AND. 1024)/1024
          C(12)=(DATAIN .AND. 2048)/2048
          RETURN

950       CONTINUE
          WRITE (5,1950) ISTAT,ICALL
1950      FORMAT (' ERROR IN CALL: STATUS = ',I6,' FROM CALL #',I6)
          CALL EXIT
          END
C
C
C
C
          SUBROUTINE BAY
C
C
C         THIS ROUTINE BUILDS THE PAYLOAD BAY ON THE PS300
C         ALL DIMENSIONS ARE IN FEET.
C
          INCLUDE '[ALFANO]PROCONST.FOR/NOLIST'
C
          REAL*4 V(3),LONG,AT(3),FM(3),UP(3)
          REAL INCR,SCL
C
C
          AT(1)=0.0
          AT(2)=0.0
          AT(3)=20.0
C
          FM(1)=0.0
          FM(2)=0.0
          FM(3)=0.0
C
          UP(1)=0.0
          UP(2)=10.0
          UP(3)=0.0
C
          CALL PBEGS('PLBAY',ERR)
            CALL PSEDCL('CLIP',.TRUE.,'',ERR)
            CALL PVIEWP('',-1.0,1.0,-1.0,1.0,1.0,1.0,'',ERR)
            CALL PFOV('',35.0,1.,10000.0,'',ERR)
            CALL PLOCKA('LOOK',AT,FM,UP,'',ERR)
              V(1)=.3048
              V(2)=.3048
              V(3)=.3048
C             SCALE FROM FEET TO METERS
            CALL PSCALE('',V,'',ERR)
              V(1)=-30.
              V(2)=-12.0
              V(3)=0.0
            CALL PTRANS('',V,'',ERR)
            CALL PSECOL('',300.,1.0,'',ERR)
            CALL PINST('','PAYLOAD',ERR)
            CALL PSECOL('',180.,1.0,'',ERR)
            CALL PINST('','SHOULDER',ERR)
```

```
          CALL PSECOL('*',90.,1.0,'*',ERR)
          CALL PINST('*','BOX',ERR)
        CALL PENDS(ERR)
C
C
C         SCALE SATELLITE AND ROTATE IT TO LIE ON Z AXIS
C
        CALL PBEGS('PAYLOAD',ERR)
C          CALL PSECOL('*',240.,1.0,'*',ERR)
           V(1)=-20.0
           V(2)=0.0
           V(3)=0.0
          CALL PTRANS('*',V,'*',ERR)
           V(1)=0.0
           V(2)=0.0
           V(3)=0.0
          CALL PTRANS('TRAN',V,'*',ERR)
          CALL PROTZ('*',90.,'*',ERR)
           V(1)=.5
           V(2)=V(1)
           V(3)=V(1)
          CALL PSCALE('*',V,'*',ERR)
          DO 335 LONG=10,360,10
            CALL PROTX('*',LONG,'RTDRS',ERR)
335       CONTINUE
        CALL PENDS(ERR)
C
C         BUILD A CYLINDER
C
        CALL PBEGS('CYLINDER',ERR)
          DO 333 LONG=36,360,36
            CALL PROTY('*',LONG,'RCYL',ERR)
333       CONTINUE
        CALL PENDS(ERR)
C
C         BUILD A BOX
C
        CALL PBEGS('BOX',ERR)
           V(1)=0.0
           V(3)=0.0
           V(2)=4.0
          CALL PTRANS('*',V,'RECT',ERR)
           V(2)=6.0
          CALL PTRANS('*',V,'RECT',ERR)
           V(2)=8.0
          CALL PTRANS('*',V,'RECT',ERR)
           V(2)=10.0
          CALL PTRANS('*',V,'RECT',ERR)
           V(2)=12.0
          CALL PTRANS('*',V,'RECT',ERR)
        CALL PENDS(ERR)
C
C         BUILD A RECTANGLE
C
        CALL PBEGS('RECT',ERR)
           V(1)=30.
           V(2)=0.
           V(3)=9.0
          CALL PSCALE('*',V,'SQUARE',ERR)
        CALL PENDS(ERR)
```

```
C
C         CALL NEEDED VECTOR LISTS
C
      CALL VECTOR ('RCYL',4)
      CALL VECTOR ('RTORS',5)
      CALL VECTOR ('SQUARE',6)
C
      INCR=.3
      SCL=-5.0
C
C
C   BUILD A WRIST
C
      CALL PBEGS('WRIST',ERR)
        CALL PROTX('PITCH',0.0,'',ERR)
          V(1)=.5415
          V(2)=1.23
          V(3)=.5415
        CALL PSCALE('',V,'CYLINDER',ERR)
          V(1)=0.0
          V(2)=1.23
          V(3)=0.0
        CALL PTRANS('',V,'',ERR)
        CALL PROTZ('YAW',0.0,'',ERR)
        CALL PROTY('ROLL',0.0,'',ERR)
          V(1)=.5415
          V(2)=4.93
          V(3)=.5415
        CALL PSCALE('',V,'CYLINDER',ERR)
      CALL PENDS(ERR)
C
C   BUILD A FOREARM (W/ ELBOW)
C
      CALL PBEGS('ELBOW',ERR)
          V(1)=0.0
          V(2)=0.0
          V(3)=.5415
        CALL PTRANS('',V,'',ERR)
        CALL PROTX('PITCH',0.0,'',ERR)
          V(1)=0.0
          V(2)=0.0
          V(3)=-.5415
        CALL PTRANS('',V,'',ERR)
          V(1)=.5415
          V(2)=23.1625
          V(3)=.5415
        CALL PSCALE('',V,'CYLINDER',ERR)
          V(1)=0.0
          V(2)=23.1625
          V(3)=0.0
        CALL PTRANS('',V,'WRIST',ERR)
      CALL PENDS(ERR)
C
C   BUILD THE SHOULDER
C
      CALL PBEGS('SHOULDER',ERR)
        CALL PROTX('',-90.,'',ERR)
          V(1)=25.0
          V(2)=-7.5
          V(3)=11.2
```

```
          CALL PTRANS('TRAN',V,'',ERR)
          CALL PROTZ('',90.0,'',ERR)
          CALL PROTY('',180.0,'',ERR)
          CALL PROTZ('YAW',0.0,'',ERR)
            V(1)=0.0
            V(2)=0.0
            V(3)=-0.001
          CALL PTRANS('',V,'',ERR)
          CALL PROTX('PITCH',0.0,'',ERR)
            V(1)=.5415
            V(2)=20.921
            V(3)=.5415
          CALL PSCALE('',V,'CYLINDER',ERR)
            V(1)=0.0
            V(2)=20.921
            V(3)=0.0
          CALL PTRANS('',V,'ELBOW',ERR)
        CALL PENDS(ERR)
C
C
C   COMMANDS FOR WRIST ROLL
C
C         SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION
C
        CALL PFN('WRROLL','YROTATE',ERR)
        CALL PSNST('WR ROLL',1,'DLABEL1',ERR)
C
C         CONNECT INPUTS TO ACCUMULATOR
C
        CALL PFN('ACC1','ACCUMULATE',ERR)
C
        CALL PCONN('DIALS',1,1,'ACC1',ERR)
        CALL PSNREA(0.0,2,'ACC1',ERR)
        CALL PSNREA(INCR,3,'ACC1',ERR)
        CALL PSNREA(SCL,4,'ACC1',ERR)
        CALL PSNREA(447.0,5,'ACC1',ERR)
        CALL PSNREA(-447.0,6,'ACC1',ERR)
C
C         CONNECT ACCUMULATOR OUTPUT TO ROTATE FUNCTION AND THEN TO PROGRAM
C
        CALL PCONN('ACC1',1,1,'WRROLL',ERR)
        CALL PCONN('WRROLL',1,1,'WRIST.ROLL',ERR)
C
C
C   COMMANDS FOR WRIST YAW
C
C         SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION
C
        CALL PFN('WRYAW','ZROTATE',ERR)
        CALL PSNST('WR YAW',1,'DLABEL2',ERR)
C
C         CONNECT INPUTS TO ACCUMULATOR
C
        CALL PFN('ACC2','ACCUMULATE',ERR)
C
        CALL PCONN('DIALS',2,1,'ACC2',ERR)
        CALL PSNREA(0.0,2,'ACC2',ERR)
        CALL PSNREA(INCR,3,'ACC2',ERR)
        CALL PSNREA(SCL,4,'ACC2',ERR)
        CALL PSNREA(120.0,5,'ACC2',ERR)
```

```
      CALL PSNREA(-120.0,6,'ACC2',ERR)
C
C        CONNECT ACCUMULATOR OUTPUT TO ROTATE FUNCTION AND THEN TO PROGRAM
C
      CALL PCONN('ACC2',1,1,'WRYAW',ERR)
      CALL PCONN('WRYAW',1,1,'WRIST.YAW',ERR)
C
C
C  COMMANDS FOR WRIST PITCH
C
C        SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION
C
      CALL PFN('WRPITCH','XROTATE',ERR)
      CALL PSNST('WR PITCH',1,'DLABEL3',ERR)
C
C        CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC3','ACCUMULATE',ERR)
C
      CALL PCONN('DIALS',3,1,'ACC3',ERR)
      CALL PSNREA(0.0,2,'ACC3',ERR)
      CALL PSNREA(INCR,3,'ACC3',ERR)
      CALL PSNREA(SCL,4,'ACC3',ERR)
      CALL PSNREA(120.0,5,'ACC3',ERR)
      CALL PSNREA(-120.0,6,'ACC3',ERR)
C
C        CONNECT ACCUMULATOR OUTPUT TO ROTATE FUNCTION AND THEN TO PROGRAM
C
      CALL PCONN('ACC3',1,1,'WRPITCH',ERR)
      CALL PCONN('WRPITCH',1,1,'WRIST.PITCH',ERR)
C
C
C  COMMANDS FOR ELBOW PITCH
C
C        SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION
C
      CALL PFN('ELPITCH','XROTATE',ERR)
      CALL PSNST('EL PITCH',1,'DLABEL4',ERR)
C
C        CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC4','ACCUMULATE',ERR)
C
      CALL PCONN('DIALS',4,1,'ACC4',ERR)
      CALL PSNREA(0.0,2,'ACC4',ERR)
      CALL PSNREA(INCR,3,'ACC4',ERR)
      CALL PSNREA(SCL,4,'ACC4',ERR)
      CALL PSNREA(160.0,5,'ACC4',ERR)
      CALL PSNREA(-2.0,6,'ACC4',ERR)
C
C        CONNECT ACCUMULATOR OUTPUT TO ROTATE FUNCTION AND THEN TO PROGRAM
C
      CALL PCONN('ACC4',1,1,'ELPITCH',ERR)
      CALL PCONN('ELPITCH',1,1,'ELBOW.PITCH',ERR)
C
C  COMMANDS FOR SHOULDER YAW
C
C        SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION
C
      CALL PFN('SHYAW','ZROTATE',ERR)
```

```
      CALL PSNST('SH YAW',1,'DLABEL5',ERR)
C
C        CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC5','ACCUMULATE',ERR)
C
      CALL PCONN('DIALS',5,1,'ACC5',ERR)
      CALL PSNREA(0.0,2,'ACC5',ERR)
      CALL PSNREA(INCR,3,'ACC5',ERR)
      CALL PSNREA(SCL,4,'ACC5',ERR)
      CALL PSNREA(180.0,5,'ACC5',ERR)
      CALL PSNREA(-180.0,6,'ACC5',ERR)
C
C        CONNECT ACCUMULATOR OUTPUT TO ROTATE FUNCTION AND THEN TO PROGRAM
C
      CALL PCONN('ACC5',1,1,'SHYAW',ERR)
      CALL PCONN('SHYAW',1,1,'SHOULDER.YAW',ERR)
C
C  COMMANDS FOR SHOULDER PITCH
C
C        SEND LABEL TO DIAL AND DECLARE ROTATE FUNCTION
C
      CALL PFN('SHPITCH','XROTATE',ERR)
      CALL PSNST('SH PITCH',1,'DLABEL6',ERR)
C
C        CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC6','ACCUMULATE',ERR)
C
      CALL PCONN('DIALS',6,1,'ACC6',ERR)
      CALL PSNREA(0.0,2,'ACC6',ERR)
      CALL PSNREA(INCR,3,'ACC6',ERR)
      CALL PSNREA(SCL,4,'ACC6',ERR)
      CALL PSNREA(2.0,5,'ACC6',ERR)
      CALL PSNREA(-145.0,6,'ACC6',ERR)
C
C        CONNECT ACCUMULATOR OUTPUT TO ROTATE FUNCTION AND THEN TO PROGRAM
C
      CALL PCONN('ACC6',1,1,'SHPITCH',ERR)
      CALL PCONN('SHPITCH',1,1,'SHOULDER.PITCH',ERR)
C
C  COMMANDS FOR SATELLITE TRANSLATION
C
C        SEND LABEL TO DIAL AND DECLARE YVECTOR FUNCTION
C
      CALL PFN('SATRAN','YVECTOR',ERR)
      CALL PSNST('SAT TRAN',1,'DLABEL7',ERR)
C
C        CONNECT INPUTS TO ACCUMULATOR
C
      CALL PFN('ACC7','ACCUMULATE',ERR)
C
      CALL PCONN('DIALS',7,1,'ACC7',ERR)
      CALL PSNREA(0.0,2,'ACC7',ERR)
      CALL PSNREA(.05,3,'ACC7',ERR)
      CALL PSNREA(2.0,4,'ACC7',ERR)
      CALL PSNREA(100.0,5,'ACC7',ERR)
      CALL PSNREA(0.0,6,'ACC7',ERR)
C
C        CONNECT ACCUMULATOR OUTPUT TO YVECTOR FUNCTION
```

```
C          AND THEN TO PROGRAM
C
      CALL PCONN('ACC7',1,1,'SATRAN',ERR)
      CALL PCONN('SATRAN',1,1,'PAYLOAD_TRAN',ERR)
C
      END
```

```
C ***************************************************************************
C <<< LPAIO >>>> LPA11-K I/O ROUTINES; J.M. LIND; 03 MAY 83; REV C
C ***************************************************************************
C
      SUBROUTINE LPAIO (MODE,IUNIT,IFLAG,DRATE,ICHAN,NCHAN,NFRAME,IOBUF,
     1 ARATE,ISTAT,ICALL,LSTAT,ISMODE)
C
C REV A DESIGNED TO USE AST CALLS AT COMPLETION OF SWEEP (FILE NAME LPAIOA.
C REV B MODIFIED TO USE EVENT FLAGS INSTEAD OF AST CALLS 03 FEB 83 (JML)
C REV C MODIFIED TO ADD I/O MODE CALLING PARAMETER "ISMODE" AND TO ALLOW
C       FOR 2 CHANNEL DIGITAL I/O WITH THE ADDITION OF MODES 7,8,17, & 18.
C
C CALLING SEPCIFICATIONS:
C
C               MODE    MODE OF CALL WITH:
C
C                       MODE  = 1 INITIALIZE LPA11-K UNIT IUNIT
C                             = 3 ANALOG INPUT
C                             = 4 ANALOG OUTPUT
C                             = 5 DIGITAL INPUT A
C                             = 6 DIGITAL OUTPUT A
C                             = 7 DIGITAL INPUT B
C                             = 8 DIGITAL OUTPUT B
C                             =13 ANALOG INPUT STATUS
C                             =14 ANALOG OUTPUT STATUS
C                             =15 DIGITAL INPUT STATUS A
C                             =16 DIGITAL OUTPUT STATUS A
C                             =17 DIGITAL INPUT STATUS B
C                             =18 DIGITAL OUTPUT STATUS B
C
C               IUNIT   UNIT NUMBER OF THE DESIRED LPA11 SUBSYSTEM:
C
C                       IUNIT = 0 USES LAA0:
C                             = 1 USES LAB0:
C
C               IFLAG   NUMBER OF THE EVENT FLAG WHICH IS TO BE SET A COMPL
C               DRATE   DESIRED SAMPLE RATE (DO NOT EXCEED 80 KHZ)
C               ICHAN   START CHANNEL NUMBER
C               NCHAN   NUMBER OF CHANNELS (MUST BE 1 FOR DIGITAL I/O)
C               NFRAME  NUMBER OF FRAMES (NCHAN PER FRAME)
C               IOBUF   BUFFER FOR DATA (NFRAME * NCHAN 2 BYTE WORDS LONG)
C
C       RETURNED INFORMATION:
C
C               ARATE   ACTUAL SAMPLE RATE USED (0 FOR ERROR)
C               ISTAT   THREE WORD ARRAY WITH:
C
C                       ISTAT = 0  ERROR IN CALL
C                             = 1  SUCCCESFUL
C                             = X  VMS ERROR CODE
```

```
C               LSTAT   INTEGER*1 (BYTE) VARIABLE USED WITH ISTAT TO DEFIN
C
C                       ISTAT   LSTAT   MEANING
C                       ------- ------- -------
C                       0       0       NORMAL - BUFFER 0 DONE
C                       -1      1       SWEEP TERMINATED OK
C                       -1      X       X = LPA11 ERROR CODE (USER'S GD PG
C
C               ICALL   CALL NUMBER OF THIS PROGRAM (RELATES TO
C                       LPA11 I/O FUNCTION WHICH WAS LAST USED BEFORE
C                       RETURN TO THE CALLING PROGRAM)
C                       IF ICALL = 0, THEN MODE IS UNDEFINED!
C
C               ISMODE  SPECIFY MODE OF LPA11 SWEEP
C
C
C NOTES:        FOR MODES 5 - 8 (DIGITAL I/O), NCHAN MUST BE 1.
C               CHANNEL NUMBERING ALWAYS STARTS WITH 0.
C               IN MODE 1, THE SAMPLE RATE OF THE LPA11 CLOCK
C               IS SET, AND THE SAME RATE IS USED ON ALL LPA11 FUNCTIONS.
C               DRATE MUST NOT EXCEED 80 KHZ.  HOWEVER, THE LPA11 USER'S
C               MANUAL SPECIFIES MAXIMUM AGGREGRATE THROUGHPUT FOR MULTIREC
C               ACTIVITIES AT 15 KHZ FOR ALL OPERATIONS COMBINED. (PARA 2.4
C
C WARNINGS:     WE SPECIFYING ISMODE = 512 IN THE DIGITAL INPUT MODE,
C               ONLY ONE CHANNEL OF DIGITAL I/O CAN BE USED AT A TIME.
C               OTHERWISE, THE PROGRAM WILL HANG WAITING FOR THE INPUT
C               FLAG TO BE SET BY THE LPA11.  WHEN USING ONLY ONE
C               CHANNEL OF DIGITAL I/O, THE ISMODE = 512 WILL WORK
C               PROPERLY.  IF ISMODE = 0 IS SPECIFIED FOR BOTH DIGITAL
C               INPUT CHANNELS, THEN TWO CHANNELS MAY BE USED AT THE
C               SAME TIME.
C
C               WHEN USING A/D OR D/A MODE, YOU MUST SPECIFY AN ISMODE
C               OF 64 IN ORDER TO USE THE MULTIREQUEST MICROCODE WHICH
C               IS LOADED BY THIS ROUTINE.
C
C
C       1.  IN THE CASE OF DIGITAL OUTPUT, THE MODE OF THE LPA11 IN
C RUNNING THE DR11-K IS TO START OUTPUT IMMEDIATELY (THE MODE SPEICIFED
C IN THE CALL SHOULD BE ISMODE = 0.)
C
C       2.  IN THE CASE OF DIGITAL INPUT, THE MODE OF THE LPA11 IN
C RUNNING THE DR11-K IS TO START INPUT ON EXTERNAL TRIGGER (THE MODE
C SPECIFIED IN THE CALL SHOULD BE ISMODE = 512).  THE "EXTERNAL"
C TRIGGER IS ACTUALLY THE DR11-K "EXTERNAL DATA READY" LINE FOR THE
C EXTERNAL DEVICE.  (SEE DR11-K TIMING DIAGRAM ON PAGE 4-7 OF THE
C DR11-K INTERFACE USER'S GUIDE AND MAINTENANCE MANUAL.)
C IN THIS MODE, INTERRUPT WILL OCCURR ONLY AFTER THE EXTERNAL DEVICE
C CYCLES THE "EXTERNAL DATA READY" LINE.
C
C       3.  NOTE THAT THE CONFIGURATION OF THE DR11-K JUMPERS IS VERY
C INPORTANT TO PROPER OPERATION OF THE DR11-K.  IN PARTICULAR, ALL
C S1 AND S2 SWITCHES SHOULD BE OFF TO DISABLE INTERRUPT BY TRANSITION
C OF THE DATA BITS (SEE TABLE 5-3).  IN ADDITION, JUMPERS W5 - W20
C MUST BE IN POSITION "B" IN ORDER TO READ DATA DIRECT FROM THE DATA
C INPUT LINES (AS OPPOSED TO THE BUFFER REGISTER INPUT).  THIS IS TUR
C DUE TO THE FACT THAT IN "BUFFER REGISTER" MODE, THE INDIVIDUAL
C DATA BITS IN THE BUFFER ARE SET ONLY ON TRANSITION OF THE DATA LINE
C (SEE PARAGRAPH 4-6 OF DR11-K INTERFACE USER'S GUIDE).  SINCE
```

```
C ALL SWITCHES ON S1 AND S2 ARE OFF, THE STATE OF JUMPERS W1-W4 IS A
C DON'T CARE.  JUMPERS W21-W23 SHOULD BE SET FOR APPROPRIATE POLARITY
C OF THE INTERNAL DATA ACCEPT AND INTERNAL DATA READY LINES.
C
C       4.  REMBER THAT ON DIGITAL OUTPUT, THIS PROGRAM SPECIFIES AT
C LEAST A 150 MICROSECOND DELAY BEFORE OUTPUT OF THE FIRST DIGITAL WORD
C (SEE PAGE 2-14 OF THE LPA11 USER'S GUIDE).  THIS IS NECESSARY
C IN ORDER TO ALLOW TIME FOR THE LPA11 TO RETRIEVE DATA FROM
C MEMORY.
```

```
C
C PARAMETER USAGE
C
C MODE   IUNIT   IFLAG   DRATE   ICHAN   NCHAN   NFRAME   IOBUF   ARATE   S
C ----   -----   -----   -----   -----   -----   ------   -----   -----   -
C  1     X               X                                       X       X
C  3     X       X               X       X       X        X               X
C  4     X       X               X       X       X        X               X
C  5     X       X               X       X       X        X               X
C  6     X       X               X       X       X        X               X
C  7     X       X               X       X       X        X               X
C  8     X       X               X       X       X        X               X
C 13                                                                      X
C 14                                                                      X
C 15                                                                      X
C 16                                                                      X
C 17                                                                      X
C 18                                                                      X
C
C ***** PROGRAM DECLARATIONS *****
C
C VARIABLE DEFINITION SECTION
C
C       IOBUF   DATA BUFFER AREA (INTEGER*2)
C       XYBUF   LPA11 CONTROL BLOCK (50 LONGWORDS)
C       XYSTAT  LPA11 COMPLETION STATUS (FORM LPA$IGTBUF CALL)
C       IFLAG   LPA11 FLAG TO BE SET AT COMPLETION OF SWEEP
C       XYIOSB  I/O STATUS BLOCK FOR LPA11 (4 WORDS)
C       DXIOSZ  I/O STATUS BLOCK FOR DIGITAL I/O TO BUFFER Z (CH A OR B)
C       XYMSKB  LPA11 SUBSYSTEM MASKS AND NUM BUFFER
C       ISTAT   LPA11 STATUS LONGWORD
C       LSTAT   LPA11 I/O COMPLETION STATUS BYTE
C       NBUF    NUMBER OF BUFFERS TO BE FILLED (LONGWORD)
C
C WHERE "XY" IS AD FOR ANALOG-TO-DIGITAL
C               DA FOR DIGITAL-TO-ANALOG
C               DI FOR DIGITAL INPUT
C               DO FOR DIGITAL OUTPUT
C
C VARIABLE TYPE SPECIFICATIONS
C
        REAL            LPA$XRATE
        INTEGER*4       SYS$CLREF
C
        INTEGER*2       IOBUF(1),ADIOSB(4),DAIOSB(4)
        INTEGER*2       DIIOSA(4),DOIOSA(4),DIIOSB(4),DOIOSB(4)
        INTEGER*4       ADMSKB(2),DAMSKB(2)
        INTEGER*4       DIMSKA(2),DOMSKA(2),DIMSKB(2),DOMSKB(2)
        INTEGER*4       ISTAT,BUFNUM,NBUF,IFLAG
        BYTE            IDSC,IEMC,LSTAT
        INTEGER*2       IDSW,IEMW
C
C SET AREA FOR CONTROL
        INTEGER*4       ADBUF(50),DABUF(50)
        INTEGER*4       DIBUFA(50),DOBUFA(50),DIBUFB(50),DOBUFB(50)
        EQUIVALENCE     (ADIOSB(1),ADBUF(1)),(DAIOSB(1),DABUF(1))
        EQUIVALENCE     (DIIOSA(1),DIBUFA(1)),(DOIOSA(1),DOBUFA(1))
        EQUIVALENCE     (DIIOSB(1),DIBUFB(1)),(DOIOSB(1),DOBUFB(1))
C
C
```

```
C ***** START OF PROGRAM *****
C DETERMINE MODE
C
C       LSTAT = 0
        GO TO (100,50,300,400,500,600,700,800,50,50,50,50,
     1 1300,1400,1500,1600,1700,1800),MODE
C EXECUTION STARTS HERE IF MODE IS UNDEFINED
 50     ICALL = 0
        ISTAT = 0
        GO TO 1950
C
C
C ***** MODE = 1 *****
C LOAD LPA11 SPECIFIED BY IUNIT WITH MICROCODE FOR MULTIREQUEST MODE
C
 100    CONTINUE
        ICALL = 101
        CALL LPA$LOADMC (1,IUNIT,ISTAT,IERROR)
        IF (.NOT. ISTAT) GO TO 1950
C
C USE XRATE ROUTINE TO CALCULATE RATE AND PRESET VALUES FOR CLOCK A
C
C RATES ARE SUPPLIED/RETURNED BUT LPA$XRATE REQUIRES INTERVALS
        AINTRVL = 1./DRATE
        ICALL = 102
        ACTUAL = LPA$XRATE (AINTRVL,IRATE,IPRSET,0)
        ARATE  = 1./ACTUAL
C
C SET CLOCK RATE TO SPECIFIEC SAMPLE RATE (DO NOT EXCEED ABOUT 80 KHZ)
C
        ICALL = 103
        CALL LPA$CLOCKA (IRATE,IPRSET,ISTAT,IUNIT)
        GO TO 1950
```

```
C ***** MODE = 3 *****
C START ANALOG-TO-DIGITAL INPUT SWEEP
C
 300    CONTINUE
C
C CLEAR A/D EVENT FLAG
        ICALL = 300
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C INITIALIZE ADBUF ARRAY FOR SWEEP
C
        ICALL = 301
        CALL LPA$SETIBF (ADBUF,ISTAT,ADMSKB,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
        ICALL = 302
        CALL LPA$LAMSKS (ADMSKB,IUNIT)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 303
        CALL LPA$RLSBUF (ADBUF,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C START A/D SWEEP BY SPECIFYING ONLY ONE BUFFER
C
        NPOINT = NFRAME * NCHAN
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 304
        CALL LPA$ADSWP (ADBUF,NPOINT,NBUF,ISMODE,,%VAL(IFLAG),,ICHAN,
     1 NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF A/D SWEEP CALL
        GO TO 1950
C
```

```
C ***** MODE = 4 *****
C START DIGITAL-TO-ANALOG OUTPUT SWEEP
C
 400    CONTINUE
C
C CLEAR D/A EVENT FLAG
        ICALL = 400
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C INITIALIZE DABUF ARRAY FOR SWEEP
C
        ICALL = 401
        CALL LPA$SETIBF (DABUF,ISTAT,DAMSKB,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
        ICALL = 402
        CALL LPA$LAMSKS (DAMSKB,IUNIT)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 403
        CALL LPA$RLSBUF (DABUF,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C CALCULATE NUMBER OF DATA POINTS
        NPOINT = NFRAME * NCHAN
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C IN D/A MODE, A DELAY OF A LEAST 150 MICROSECONDS MUST BE SPECIFIED BEFORE
C THE FIRST CONVERSION TAKES PLACE.  SINCE THE LPA$XRATE CALL RETURNS THE V
C OF IRATE (SPECIFYING A CLOCK RATE), LDELAY (THE DELAY IN IRATE
C UNITS BEFORE FIRST SAMPLE) IS SET.  (PARA 2.4.1 OF LPA11 USER'S GUIDE)
C IRATE = 1 FOR 1 MHZ; IRATE = 2 FOR 100 KHZ; IRATE = 3 FOR 10KHZ; ETC.
        LDELAY = 1
        IF (IRATE.EQ.1) LDELAY = 150
        IF (IRATE.EQ.2) LDELAY = 15
        IF (IRATE.EQ.3) LDELAY = 2
C SPECIFY SAMPLE ON EVERY CLOCK OVERFLOW
        IDWELL = 1
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 404
        CALL LPA$DASWP (DABUF,NPOINT,NBUF,ISMODE,IDWELL,%VAL(IFLAG),
     1 LDELAY,ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF D/A SWEEP CALL
        GO TO 1950
```

```
C ***** MODE = 5 *****
C START DIGITAL INPUT SWEEP FOR "CHANNEL A"
C
 500    CONTINUE
C
C CLEAR DIGITAL INPUT EVENT FLAG
        ICALL = 500
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
        ISTAT = 0
        ICALL = 501
        IF (NCHAN.NE.1) GO TO 1950
        ISTAT = 1
C
C INITIALIZE DIBUFA ARRAY FOR SWEEP
C
        ICALL = 502
        CALL LPA$SETIBF (DIBUFA,ISTAT,DIMSKA,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
        IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
        IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
        IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
        IEMW = -1
        ICALL = 503
        CALL LPA$LAMSKS (DIMSKA,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 504
        CALL LPA$RLSBUF (DIBUFA,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C START DIGITAL INPUT SWEEP BY SPECIFYING ONLY ONE BUFFER
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
        NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 505
        CALL LPA$DISWP (DIBUFA,NPOINT,NBUF,ISMODE,,%VAL(IFLAG),,
     1 ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL INPUT SWEEP CALL
        GO TO 1950
```

```
C ***** MODE = 6 *****
C START DIGITAL OUTPUT SWEEP FOR "CHANNEL A"
C
 600    CONTINUE
C
C CLEAR DIGITAL OUTPUT EVENT FLAG
        ICALL = 600
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
        ISTAT = 0
        ICALL = 601
        IF (NCHAN.NE.1) GO TO 1950
        ISTAT = 1
C
C INITIALIZE DOBUFA ARRAY FOR SWEEP
C
        ICALL = 602
        CALL LPA$SETIBF (DOBUFA,ISTAT,DOMSKA,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
        IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
        IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
        IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
        IEMW = -1
        ICALL = 603
        CALL LPA$LAMSKS (DOMSKA,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 604
        CALL LPA$RLSBUF (DOBUFA,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
        NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C
C IN DO MODE, A DELAY OF A LEAST 150 MICROSECONDS MUST BE SPECIFIED BEFORE
C THE FIRST CONVERSION TAKES PLACE.  SINCE THE LPA$XRATE CALL RETURNS THE
C OF IRATE (SPECIFYING A CLOCK RATE), LDELAY (THE DELAY IN IRATE
C UNITS BEFORE FIRST SAMPLE) IS SET.  (PARA 2.4.1 OF LPA11 USER'S GUIDE)
C IRATE = 1 FOR 1 MHZ; IRATE = 2 FOR 100 KHZ; IRATE = 3 FOR 10KHZ; ETC.
        LDELAY = 1
        IF (IRATE.EQ.1) LDELAY = 150
        IF (IRATE.EQ.2) LDELAY = 15
        IF (IRATE.EQ.3) LDELAY = 2
C SPECIFY SAMPLE ON EVERY CLOCK OVERFLOW
        IDWELL = 1
C PROCEED WITH SWEEP START CALL
```

```
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 605
        CALL LPA$DOSWP (DOBUFA,NPOINT,NBUF,ISMODE,IDWELL,%VAL(IFLAG),
     1 LDELAY,ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL OUTPUT SWEEP CAL
        GO TO 1950
```

```
C ***** MODE = 7 *****
C START DIGITAL INPUT SWEEP FOR "CHANNEL B"
C
 700    CONTINUE
C
C CLEAR DIGITAL INPUT EVENT FLAG
        ICALL = 700
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
        ISTAT = 0
        ICALL = 701
        IF (NCHAN.NE.1) GO TO 1950
        ISTAT = 1
C
C INITIALIZE DIBUFB ARRAY FOR SWEEP
C
        ICALL = 702
        CALL LPA$SETIBF (DIBUFB,ISTAT,DIMSKB,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
        IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
        IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
        IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
        IEMW = -1
        ICALL = 703
        CALL LPA$LAMSKS (DIMSKB,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 704
        CALL LPA$RLSBUF (DIBUFB,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C START DIGITAL INPUT SWEEP BY SPECIFYING ONLY ONE BUFFER
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
        NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 705
        CALL LPA$DISWP (DIBUFB,NPOINT,NBUF,ISMODE,,%VAL(IFLAG),,
     1 ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C
C
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL INPUT SWEEP CALL
        GO TO 1950
```

```
C ***** MODE = 8 *****
C START DIGITAL OUTPUT SWEEP FOR "CHANNEL B"
C
 800    CONTINUE
C
C CLEAR DIGITAL OUTPUT EVENT FLAG
        ICALL = 800
        ISTAT = SYS$CLREF (%VAL(IFLAG))
        IF (.NOT. ISTAT) GO TO 1950
C
C CHECK THAT NCHAN IS EQUAL TO ONE
        ISTAT = 0
        ICALL = 801
        IF (NCHAN.NE.1) GO TO 1950
        ISTAT = 1
C
C INITIALIZE DOBUFB ARRAY FOR SWEEP
C
        ICALL = 802
        CALL LPA$SETIBF (DOBUFB,ISTAT,DOMSKB,IOBUF)
        IF (.NOT. ISTAT) GO TO 1950
C
C SET UP FOR LPA11 SUBSYSTEM NUMBER
C
C SPECIFY START WORD CHANNEL OF CHANNEL ZERO (I/O GUIDE PAGE 5-22)
        IDSC = 0
C SPECIFY EVENT MARK WORD CHANNEL OF CHANNEL 0
        IEMC = 0
C SPECIFY DIGITAL START WORD MASK OF ALL BITS
        IDSW = -1
C SPCEIFY EVENT MARK WORD MASK OF ALL BITS
        IEMW = -1
        ICALL = 803
        CALL LPA$LAMSKS (DOMSKB,IUNIT,,IDSC,IEMC,IDSW,IEMW,)
C
C RELEASE THE BUFFER (BUFFER NUMBERS ARE USED RATHER THAN NAMES)
C
        ICALL = 804
        CALL LPA$RLSBUF (DOBUFB,ISTAT,0)
        IF (.NOT. ISTAT) GO TO 1950
C
C FOR THIS MODE, THE NUMBER OF POINTS MUST EQUAL NUMBER OF FRAMES
        NPOINT = NFRAME
C SPECIFY ONLY ONE BUFFER TO BE FILLED
        NBUF = 1
C IN REV C OF THIS ROUTINE, ISMODE IS SPECIFIED IN THE CALLING PARAMETERS
C
C IN DO MODE, A DELAY OF A LEAST 150 MICROSECONDS MUST BE SPECIFIED BEFORE
C THE FIRST CONVERSION TAKES PLACE.  SINCE THE LPA$XRATE CALL RETURNS THE V
C OF IRATE (SPECIFYING A CLOCK RATE), LDELAY (THE DELAY IN IRATE
C UNITS BEFORE FIRST SAMPLE) IS SET.  (PARA 2.4.1 OF LPA11 USER'S GUIDE)
C IRATE = 1 FOR 1 MHZ; IRATE = 2 FOR 100 KHZ; IRATE = 3 FOR 10KHZ; ETC.
        LDELAY = 1
        IF (IRATE.EQ.1) LDELAY = 150
        IF (IRATE.EQ.2) LDELAY = 15
        IF (IRATE.EQ.3) LDELAY = 2
C SPECIFY SAMPLE ON EVERY CLOCK OVERFLOW
        IDWELL = 1
```

```
C PROCEED WITH SWEEP START CALL
C SWEEP CALL SPECIFIES FLAG BE SET AT COMPLETION
        ICALL = 805
        CALL LPA$DOSWP (DOBUFB,NPOINT,NBUF,ISMODE,IDWELL,%VAL(IFLAG),
     1 LDELAY,ICHAN,NCHAN,ISTAT)
        IF (.NOT. ISTAT) GO TO 1950
C                                                                    CALL
C RETURN TO CALLING PROGRAM ... ISTAT IS STATUS OF DIGITAL OUTPUT SWEEP CAL
        GO TO 1950
```

```
C ***** MODE = 13 *****
C GET STATUS OF A/D SWEEP
C
 1300    CONTINUE
C
         ICALL = 1301
         ISTAT = LPASIWTBUF(ADBUF)
         LSTAT = IAND(ADIOSB(3),'FF00'X)/256
         GO TO 1950
C
C
C ***** MODE = 14 *****
C GET STATUS OF D/A SWEEP
C
 1400    CONTINUE
C
         ICALL = 1401
         ISTAT = LPASIWTBUF(DABUF)
         LSTAT = IAND(DAIOSB(3),'FF00'X)/256
         GO TO 1950
C
C
C ***** MODE = 15 *****
C GET STATUS OF DIGITAL INPUT SWEEP FOR "CHANNEL A"
C
 1500    CONTINUE
C
         ICALL = 1501
         ISTAT = LPASIWTBUF(DIBUFA)
         LSTAT = IAND(DIIOSA(3),'FF00'X)/256
         GO TO 1950
C
C
C ***** MODE = 16 *****
C GET STATUS OF DIGITAL OUTPUT SWEEP FOR "CHANNEL A"
C
 1600    CONTINUE
C
         ICALL = 1601
         ISTAT = LPASIWTBUF(DOBUFA)
         LSTAT = IAND(DOIOSA(3),'FF00'X)/256
         GO TO 1950
C
C
C ***** MODE = 17 *****
C GET STATUS OF DIGITAL INPUT SWEEP FOR "CHANNEL B"
C
 1700    CONTINUE
C
         ICALL = 1701
         ISTAT = LPASIWTBUF(DIBUFB)
         LSTAT = IAND(DIIOSB(3),'FF00'X)/256
         GO TO 1950
C
C
C ***** MODE = 18 *****
C GET STATUS OF DIGITAL OUTPUT SWEEP FOR "CHANNEL B"
C
 1800    CONTINUE
C
```

```
            ICALL = 1801
            ISTAT = LPA$IWTBUF(DOBUFB)
            LSTAT = IAND(DOIOSB(3),'FF00'X)/256
            GO TO 1950
C
C
C
C ***** ERROR SERVICE ROUTINE *****
C
 1950   CONTINUE
C TRANSFER STATUS INFORMATION ON CALLING PARAMETER
C
            RETURN
C
            END
```

```
C
C
C
C    THE FOLLOWING ERROR HANDLER DEMONSTRATES THE GENERAL
C    OVERALL RECOMMENDED FORM THAT THE USER'S OWN ERROR
C    HANDLER SHOULD FOLLOW.
C
C    THIS ERROR HANDLER UPON BEING INVOKED WRITES ALL
C    MESSAGES TO THE DATA FILE:  'PROERROR.LOG'.  ERROR
C    AND WARNING EXPLANATION MESSAGES ARE ARE WRITTEN TO
C    A DATA FILE FOR 2 REASONS:
C
C
C        1.  THE ERROR HANDLER SHOULD NOT IMMEDIATELY
C            WRITE INFORMATION OUT ON THE PS 300 SCREEN
C            SINCE THE EXPLANATORY TEXT DEFINING THE ERROR
C            OR WARNING CONDITION MAY BE TAKEN AS DATA BY
C            THE PS 300 AND THEREFORE WIND UP NOT BEING
C            DISPLAYED ON THE PS 300 SCREEN (AS IN THE
C            CASE OF A CATASTROPHIC DATA TRANSMISSION
C            ERROR).
C
C        2.  THE LOGGING OF ERRORS AND WARNINGS TO A
C            LOGFILE ALLOWS ANY ERRORS AND/OR WARNINGS
C            TO BE REVIEWED AT A LATER TIME.
C
C
C



      SUBROUTINE ERR ( ERRCOD )

C
C       PROCEDURAL INTERFACE ERROR HANDLER:
C

      INCLUDE    'PROCONST.FOR/NOLIST'
      INTEGER*4  ERRCOD
      INTEGER*4  PSVMSERR
      LOGICAL    FILOPN
      DATA       FILOPN /.FALSE./
      EXTERNAL   PSVMSERR, DETERH, PIDCOD

      IF (FILOPN) GOTO 1
C
C       OPEN ERROR FILE FOR LOGGING OF ERRORS:
C
        OPEN (UNIT=10, FILE='PROERROR.LOG', STATUS='NEW',
     &        DISP='KEEP', ORGANIZATION='SEQUENTIAL',
     &        ACCESS='SEQUENTIAL', CARRIAGECONTROL='LIST')
        FILOPN = .TRUE.
C     END IF
    1 CALL PIDCOD (ERRCOD)
      IF (ERRCOD .LT. 512) GOTO 3
        WRITE (10, *) 'PS-I-ATDCOMLNK:  ATTEMPTING TO '
     &              // 'DETACH PS 300/HOST COMMUNICATIONS '
     &              // 'LINK.'
C
C       WHEN WE ATTEMPT TO PERFORM THE DETACH, USE A
```

```
C         DIFFERENT ERROR HANDLER SO AS NOT TO GET CAUGHT
C         IN A RECURSIVE LOOP IF WE CONSISTENTLY GET AN
C         ERROR WHEN ATTEMPTING TO DETACH.
C
          CALL PDTACH (DETERH)
          CLOSE (UNIT=10)
          IF ((ERRCOD .LT. PSFPAF) .OR.
     &        (ERRCOD .GT. PSFPPF)) GOTO 2
C
C           IDENTIFY VMS ERROR IF THERE WAS ONE
C
            CALL LIBSSTOP (%VAL (PSVMSERR ()))
            GOTO 3
C         ELSE
        2   STOP
C         END IF
C       END IF
        3 RETURN
          END



          SUBROUTINE DETERH (ERRCOD)

C
C         MAIN ERROR HANDLER DETACH ERROR HANDLER:
C

          INTEGER*4  ERRCOD
          EXTERNAL   PIDCOD

          WRITE (10, *) 'PS-I-ERRWARDET:  ERROR/WARNING '
     &                  // 'TRYING TO DETACH '
     &                  // 'THE COMMUNICATIONS'
          WRITE (10, *) 'LINK BETWEEN THE PS 300 AND THE HOST.'
          CALL PIDCOD (ERRCOD)
          RETURN
          END



          SUBROUTINE PIDCOD (ERRCOD)

C
C           PIDCOD: IDENTIFY PROCEDURAL INTERFACE COMPLETION
C                   CODE.
C

          INCLUDE    'PROCONST.FOR/NOLIST'
          INTEGER*4  ERRCOD
          CHARACTER  VMSDEF*133, PIDEF*133
          INTEGER*4  PSVMSERR
          CHARACTER  MSSG1*55, MSSG2*67
          PARAMETER  (MSSG1 = 'PS-W-UNRCOMCOD:  PROCEDURAL '
     &                      // 'INTERFACE '
     &                      // '(GSR) COMPLETION ')
          EXTERNAL   PSVMSERR

          WRITE (10, *) 'PS-I-PROERRWAR:  PROCEDURAL '
     &                  // 'INTERFACE WARNING/'
```

```
     &            // 'ERROR COMPLETION CODE WAS '
      WRITE (10, *) 'RECEIVED.'
      IF (ERRCOD .NE. PSWBNC) GOTO 1
        WRITE (10, *) 'PS-W-BADNAMCHR:  BAD CHARACTER '
     &              // 'IN NAME WAS '
     &              // 'TRANSLATED TO:  "_".'
        GOTO 1000
C     ELSE
    1 IF (ERRCOD .NE. PSWNTL) GOTO 2
        WRITE (10, *) 'PS-W-NAMTOOLON:  NAME TOO '
     &              // 'LONG.  NAME WAS '
     &              // 'TRUNCATED TO '
        WRITE (10, *) '256 CHARACTERS.'
        GOTO 1000
C     ELSE
    2 IF (ERRCOD .NE. PSWSTL) GOTO 7
        WRITE (10, *) 'PS-W-STRTOOLON:  STRING TOO '
     &              // 'LONG.  STRING '
     &              // 'WAS TRUNCATED '
        WRITE (10, *) 'TO 240 CHARACTERS.'
        GOTO 1000
C     ELSE
    7 IF (ERRCOD .NE. PSWAAD) GOTO 8
        WRITE (10, *) 'PS-W-ATTALRDON:  ATTACH '
     &              // 'ALREADY DONE.  '
     &              // 'MULTIPLE CALL TO PATTCH WITHOUT'
        WRITE (10, *) 'INTERVENING PDTACH CALL IGNORED.'
        GOTO 1000
C     ELSE
    8 IF (ERRCOD .NE. PSWAKS) GOTO 9
        WRITE (10, *) 'PS-W-ATNKEYSEE:  ATTENTION KEY '
     &              // 'SEEN (DEPRESSED).'
        CALL PIBMSP
        GOTO 1000
C     ELSE
    9 IF (ERRCOD .NE. PSWBGC) GOTO 10
        WRITE (10, *) 'PS-W-BADGENCHR:  BAD GENERIC '
     &              // 'CHANNEL CHARACTER.  BAD '
        WRITE (10, *) 'CHARACTER IN STRING SENT VIA:  '
     &              // 'PPUTGX  WAS TRANSLATED TO '
        WRITE (10, *) 'A BLANK.'
        CALL PIBMSP
        GOTO 1000
C     ELSE
   10 IF (ERRCOD .NE. PSWBSC) GOTO 11
        WRITE (10, *) 'PS-W-BADSTRCHR:  BAD '
     &              // 'CHARACTER IN STRING WAS '
     &              // 'TRANSLATED TO A BLANK.'
        CALL PIBMSP
        GOTO 1000
C     ELSE
   11 IF (ERRCOD .NE. PSWBPC) GOTO 12
        WRITE (10, *) 'PS-W-BADPARCHR:  BAD PARSER '
     &              // 'CHANNEL CHARACTER.  BAD '
     &              // 'CHARACTER IN STRING SENT TO'
        WRITE (10, *) 'PS 300 PARSER VIA:  PPUTP  '
     &              // 'WAS TRANSLATED TO A BLANK.'
        CALL PIBMSP
        GOTO 1000
C     ELSE
```

```
   12 IF (ERRCOD .NE. PSEIMC) GOTO 13
        WRITE (10, *) 'PS-E-INVMUXCHA:  INVALID '
     &              // 'MULTIPLEXING CHANNEL '
     &              // 'SPECIFIED IN CALL TO:'
        WRITE (10, *) 'PMUXCI, PMUXP, OR PMUXG.'
        GOTO 1000
C      ELSE
   13 IF (ERRCOD .NE. PSEIVC) GOTO 14
        WRITE (10, *) 'PS-E-INVVECCLA:  INVALID '
     &              // 'VECTOR LIST CLASS '
     &              // 'SPECIFIED'
        WRITE (10, *) 'IN CALL TO:  PVCBEG.'
        GOTO 1000
C      ELSE
   14 IF (ERRCOD .NE. PSEIVD) GOTO 15
        WRITE (10, *) 'PS-E-INVVECDIM:  INVALID '
     &              // 'VECTOR LIST DIMENSION '
     &              // 'SPECIFIED IN CALL TO'
        WRITE (10, *) 'PVCBEG.'
        GOTO 1000
C      ELSE
   15 IF (ERRCOD .NE. PSEPOE) GOTO 16
        WRITE (10, *) 'PS-E-PREOPEEXP:  PREFIX '
     &              // 'OPERATOR CALL WAS '
     &              // 'EXPECTED.'
        GOTO 1000
C      ELSE
   16 IF (ERRCOD .NE. PSEFOE) GOTO 17
        WRITE (10, *) 'PS-E-FOLOPEEXP:  FOLLOW '
     &              // 'OPERATOR CALL WAS '
     &              // 'EXPECTED.'
        GOTO 1000
C      ELSE
   17 IF (ERRCOD .NE. PSELBE) GOTO 18
        WRITE (10, *) 'PS-E-LABBLKEXP:  CALL TO '
     &              // 'PLAADD OR PLAEND WAS '
     &              // 'EXPECTED.'
        GOTO 1000
C      ELSE
   18 IF (ERRCOD .NE. PSEVLE) GOTO 19
        WRITE (10, *) 'PS-E-VECLISEXP:  CALL TO '
     &              // 'PVCLIS OR PVCEND '
     &              // 'WAS EXPECTED.'
        GOTO 1000
C      ELSE
   19 IF (ERRCOD .NE. PSEAMV) GOTO 20
        WRITE (10, *) 'PS-E-ATTMULVEC:  ATTEMPTED '
     &              // 'MULTIPLE CALL '
     &              // 'SEQUENCE TO PVCLIS IS NOT'
        WRITE (10, *) 'PERMITTED FOR BLOCK '
     &              // 'NORMALIZED VECTORS.'
        GOTO 1000
C      ELSE
   20 IF (ERRCOD .NE. PSEMLB) GOTO 21
        WRITE (10, *) 'PS-E-MISLABBEG:  MISSING '
     &              // 'LABEL BLOCK BEGIN CALL.  '
     &              // 'CALL TO PLAADD OR PLAEND'
        WRITE (10, *) 'WITHOUT CALL TO:  PLABEG.'
        GOTO 1000
C      ELSE
```

```
   21 IF (ERRCOD .NE. PSEMVB) GOTO 22
        WRITE (10, *) 'PS-E-MISVECBEG:  MISSING '
     &              // 'VECTOR LIST BEGIN '
     &              // 'CALL.  CALL TO PVCLIS'
        WRITE (10, *) 'OR PVCEND WITHOUT CALL '
     &              // 'TO:  PVCBEG.'
        GOTO 1000
C      ELSE
   22 IF (ERRCOD .NE. PSENUN) GOTO 23
        WRITE (10, *) 'PS-E-NULNAM:  NULL NAME '
     &              // 'PARAMETER IS NOT ALLOWED.'
        GOTO 1000
C      ELSE
   23 IF (ERRCOD .NE. PSEBCT) GOTO 24
        WRITE (10, *) 'PS-E-BADCOMTYP:  BAD '
     &              // 'COMPARISON TYPE OPERATOR '
     &              // 'SPECIFIED IN '
        WRITE (10, *) 'CALL TO:  PIFLEV.'
        GOTO 1000
C      ELSE
   24 IF (ERRCOD .NE. PSEIFN) GOTO 25
        WRITE (10, *) 'PS-E-INVFUNNAM:  INVALID '
     &              // 'FUNCTION NAME.  '
     &              // 'ATTEMPTED PS 300'
        WRITE (10, *) 'FUNCTION INSTANCE FAILED '
     &              // 'BECAUSE THE NAMED '
     &              // 'FUNCTION CANNOT POSSIBLY'
        WRITE (10, *) 'EXIST.  THE FUNCTION NAME '
     &              // 'IDENTIFYING THE '
     &              // 'FUNCTION TYPE TO INSTANCE'
        WRITE (10, *) 'WAS LONGER THAN 256 CHARACTERS.'
        GOTO 1000
C      ELSE
   25 IF (ERRCOD .NE. PSENNR) GOTO 26
        WRITE (10, *) 'PS-E-NULNAMREQ:  NULL NAME '
     &              // 'PARAMETER IS '
     &              // 'REQUIRED IN OPERATE NODE'
        WRITE (10, *) 'CALL FOLLOWING A PPREF OR '
     &              // 'PFOLL PROCEDURE CALL.'
        GOTO 1000
C      ELSE
   26 IF (ERRCOD .NE. PSETME) GOTO 27
        WRITE (10, *) 'PS-E-TOOMANEND:  TOO '
     &              // 'MANY END_STRUCTURE CALLS '
     &              // 'INVOKED.'
        GOTO 1000
C      ELSE
   27 IF (ERRCOD .NE. PSENOA) GOTO 28
        WRITE (10, *) 'PS-E-NOTATT:  THE PS 300 '
     &              // 'COMMUNICATIONS LINK '
     &              // 'HAS NOT '
        WRITE (10, *) 'YET BEEN ESTABLISHED.  '
     &              // 'PATTCH HAS NOT BEEN '
     &              // 'CALLED OR FAILED.'
        GOTO 1000
C      ELSE
   28 IF (ERRCOD .NE. PSEODR) GOTO 29
        WRITE (10, *) 'PS-E-OVEDURREA:  AN '
     &              // 'OVERRUN OCCURRED DURING '
     &              // 'A READ OPERATION.'
```

```
          WRITE (10, *) 'THE SPECIFIED INPUT BUFFER '
     &              // 'IN CALL TO:  PGET  '
     &              // 'OR:  PGETW'
          WRITE (10, *) 'WAS TOO SMALL AND '
     &              // 'TRUNCATION HAS OCCURRED.'
          GOTO 1000
C       ELSE
     29 IF (ERRCOD .NE. PREICP) GOTO 38
     38 IF (ERRCOD .NE. PSEPDT) GOTO 39
          WRITE (10, *) 'PS-E-PHYDEVTYP:  MISSING '
     &              // 'OR INVALID PHYSICAL '
     &              // 'DEVICE TYPE'
          WRITE (10, *) 'SPECIFIER IN CALL TO PATTCH.'
          CALL PVAXSP
          GOTO 1000
C       ELSE
     39 IF (ERRCOD .NE. PSELDN) GOTO 40
          WRITE (10, *) 'PS-E-LOGDEVNAM:  MISSING '
     &              // 'OR INVALID LOGICAL '
     &              // 'DEVICE NAME'
          WRITE (10, *) 'SPECIFIER IN CALL TO PATTCH.'
          CALL PVAXSP
          GOTO 1000
C       ELSE
     40 IF (ERRCOD .NE. PSEADE) GOTO 41
          WRITE (10, *) 'PS-E-ATTDELEXP:  ATTACH '
     &              // 'PARAMETER STRING '
     &              // 'DELIMITER'
          WRITE (10, *) '"/" WAS EXPECTED.'
          CALL PVAXSP
          GOTO 1000
C       ELSE
     41 IF (ERRCOD .NE. PSFPAF) GOTO 42
          WRITE (10, *) 'PS-F-PHYATTFAI:  '
     &              // 'PHYSICAL ATTACH OPERATION '
     &              // 'FAILED.'
          GOTO 1000
C       ELSE
     42 IF (ERRCOD .NE. PSFPDF) GOTO 43
          WRITE (10, *) 'PS-F-PHYDETFAI:  PHYSICAL '
     &              // 'DETACH OPERATION '
     &              // 'FAILED.'
          GOTO 1000
C       ELSE
     43 IF (ERRCOD .NE. PSFPGF) GOTO 44
          WRITE (10, *) 'PS-F-PHYGETFAI:  PHYSICAL '
     &              // 'GET OPERATION FAILED.'
          GOTO 1000
C       ELSE
     44 IF (ERRCOD .NE. PSFPPF) GOTO 45
          WRITE (10, *) 'PS-F-PHYPUTFAI:  PHYSICAL '
     &              // 'PUT OPERATION FAILED.'
          GOTO 1000
C       ELSE
     45 IF (ERRCOD .NE. PSFBTL) GOTO 46
          WRITE (10, *) 'PS-F-BUFTOOLAR:  BUFFER '
     &              // 'TOO LARGE ERROR IN '
     &              // 'CALL TO:  PSPUT.'
          WRITE (10, *) 'THIS ERROR SHOULD NEVER '
     &              // 'OCCUR AND INDICATES A '
```

```
     &              // 'PROCEDURAL INTERFACE (GSR)'
        WRITE (10, *) 'INTERNAL VALIDITY CHECK.'
        CALL PVAXSP
        GOTO 1000
C      ELSE
   46 IF (ERRCOD .NE. PSFWNA) GOTO 47
        WRITE (10, *) 'PS-F-WRONUMARG:  WRONG '
     &              // 'NUMBER OF ARGUMENTS '
     &              // 'IN CALL TO PROCEDURAL'
        WRITE (10, *) 'INTERFACE (GSR) LOW-LEVEL '
     &              // 'I/O PROCEDURE '
     &              // '(SOURCE FILE:  PROIOLIB.MAR).'
        WRITE (10, *) 'THIS ERROR SHOULD NEVER '
     &              // 'OCCUR AND INDICATES A '
     &              // 'PROCEDURAL INTERFACE (GSR)'
        WRITE (10, *) 'INTERNAL VALIDITY CHECK.'
        CALL PVAXSP
        GOTO 1000
C      ELSE
   47 IF (ERRCOD .NE. PSFPTL) GOTO 48
        WRITE (10, *) 'PS-F-PROTOOLAR:  PROMPT '
     &              // 'BUFFER TOO LARGE '
     &              // 'ERROR IN CALL TO:  PSPRCV.'
        WRITE (10, *) 'THIS ERROR SHOULD NEVER '
     &              // 'OCCUR AND INDICATES A '
     &              // 'PROCEDURAL INTERFACE (GSR)'
        WRITE (10, *) 'INTERNAL VALIDITY CHECK.'
        CALL PVAXSP
        GOTO 1000
C      ELSE
C
C      UNKNOWN ERROR MESSAGE ERROR MESSAGE.
C
   48 IF (ERRCOD .GE. 512) GOTO 49
        MSSG2 = MSSG1 // 'WARNING'
        GOTO 51
C      ELSE
   49   IF (ERRCOD .GE. 1024) GOTO 50
          MSSG2 = MSSG1 // 'ERROR '
          GOTO 51
C        ELSE
   50     MSSG2 = MSSG1 // 'FATAL ERROR '
C        END IF
C      END IF
   51 WRITE (10, *) MSSG2
      WRITE (10, *) 'CODE IS UNRECOGNIZED.'
      WRITE (10, *) 'PROBABLE PROCEDURAL '
     &            // 'INTERFACE (GSR) INTERNAL '
     &            // 'VALIDITY CHECK ERROR.'
C      END IF
 1000 IF ((ERRCOD .LT. PSFPAF) .OR.
     &    (ERRCOD .GT. PSFPPF)) GOTO 2000
        CALL PSFVMSERR ( VMSDEF, PIDEF )
        WRITE (10, *) 'DEC VAX/VMS ERROR '
     &              // 'DEFINITION IS:'
        WRITE (10, *)  VMSDEF
        WRITE (10, *) 'PROCEDURAL INTERFACE '
     &              // '(GSR) INTERPRETATION OF '
     &              // 'DEC VAX/VMS COMPLETION CODE:'
        WRITE (10, *)  PIDEF
```

```
          WRITE (10, *) 'DEC VAX/VMS ERROR CODE '
     &                  // 'VALUE WAS: ', PSVMSERR ()
C      END IF
 2000  WRITE (10, *)
       RETURN
       END



       SUBROUTINE PIBMSP

C
C        PIBMSP:  WRITE THE "IBM VERSION SPECIFIC"
C                 MESSAGE TO THE ERROR HANDLER FILE.
C

       WRITE (10, *) 'THIS ERROR/WARNING IS '
     &            // 'APPLICABLE ONLY TO THE IBM '
     &            // 'VERSION OF THE'
       WRITE (10, *) 'PROCEDURAL INTERFACE (GSR).'
       RETURN
       END



       SUBROUTINE PVAXSP

C
C        PVAXSP:  WRITE THE "DEC VAX/VMS VERSION
C                 SPECIFIC" MESSAGE TO THE ERROR
C                 HANDLER FILE.
C

       WRITE (10, *) 'THIS ERROR/WARNING IS '
     &            // 'APPLICABLE ONLY TO THE DEC '
     &            // 'VAX/VMS VERSION OF'
       WRITE (10, *) 'THE PROCEDURAL INTERFACE (GSR).'
       RETURN
       END
```

END

DTIC

8-86